Implementation and Verification of a Polling-based MAC Layer Protocol for PLC

Project Work

by

Luis Lax Cortina

at the

Institute of the Industrial Information Technology

Zeitraum:	11.12.2009 – 12.06.2010
Hauptreferent:	Prof. Dr.-Ing. Klaus Dostert
Betreuer:	Dipl.-Ing. Michael Bauer

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Karlsruhe, July the 19th, 2010

Luis Lax Cortina

TABLE OF CONTENTS

LIST OF FIGURES

TABLE INDEX

1 INTRODUCTION

The aim of this project is to create a Polling-based MAC Layer Protocol for PLC. The first step is to look up some information about how the protocol is and then create the best design.

This protocol is inside the Datalink layer, which function is to control the flow of frames inside the network and to make it as efficient as possible.

A master/slave typology is used. There is a device (master) that indicates when a slave can transmit a frame. The main advantage of this typology is the low collision risk.

The second step is the VHDL implementation which is made in Modelsim. This program is a simulator too, so the functional correctness of the design can be checked with it. Inside this step, there are two stages: firstly the ARQ scheme is made and when it works, the rest of the design is implemented.

It is important to simulate the code because all the process, until the result is reached, can be studied, but it does not mean it works in a real device. An Altera kit with a Cyclon III FPGA is used to make the finals tests.

2 THEORETICAL CONCEPTS

2.1 INTRODUCTION

It is important to know about some concepts before explaining the Mac protocol which is included in the data link layer inside the OSI Layer Model. Moreover this is an own design, so it has several specifications which were fixed at the beginning of the project.

2.2 OSI LAYER MODEL

The model was created with the aim of standardizing communication in different networks. All the networks, which use this model, will be able to interchange data between them.

The model is divided in seven layers and each of them has one aim. The layers are: Application, Presentation, Session, Transport, Network, Data Link and Physical. The devices must use the standard of these layers to make easy the connection with others.

Every layer receives information from the earlier layer, and then the data is given to the next layer after it has been processed. The function of each layer is different and depends on the direction of transmission. If it is in the transmitter part, the layer receives the frame from the top layer and adds its header. After that, the new frame is given to the layer bellow it. Meanwhile in the receiver, the layer receives the frame from the bottom layer and gives it to the top layer. Although the transfer of information is vertical, the aim of this model is to transmit information between the same layers, it means horizontal transmission.

The flow of information between two systems which have all the layers is shown in the fig 2.1. It is not compulsory every device have all the layers, in fact most of them do not have the top layers.

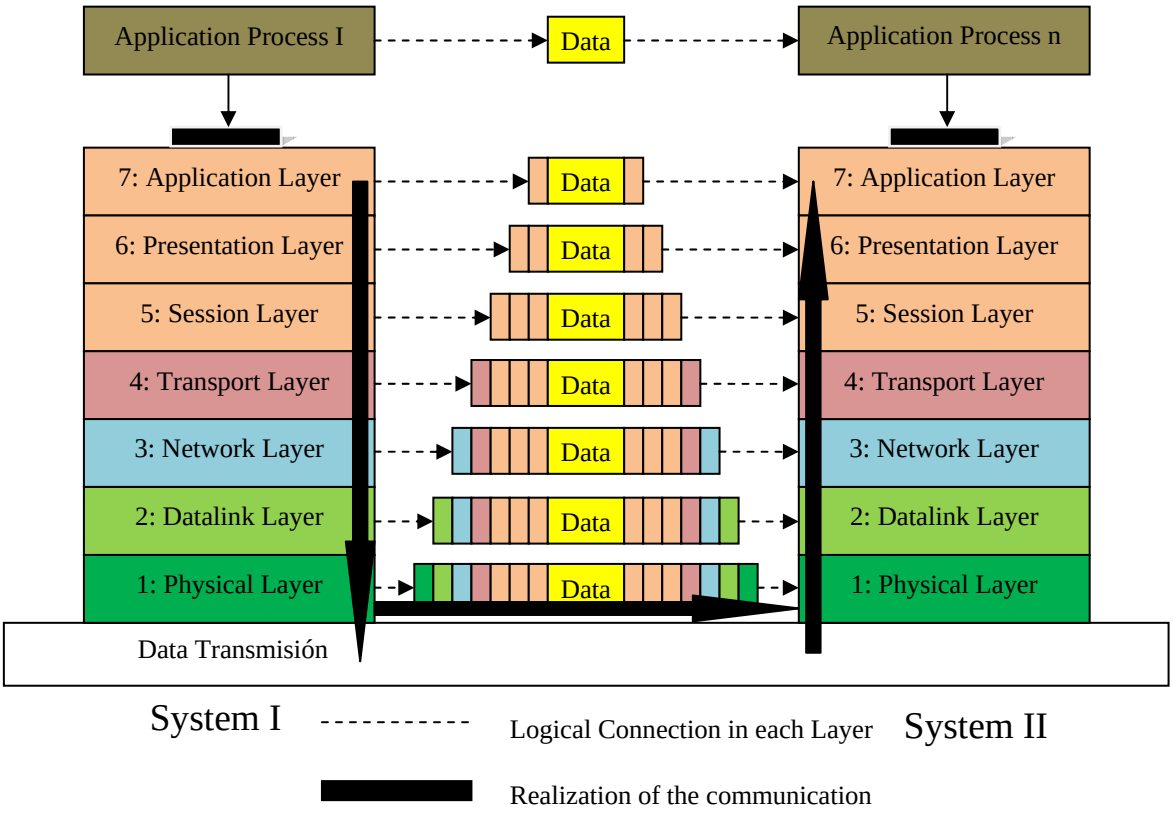

Fig 2.1 OSI Model Layer

The MAC protocol is inside the Datalink layer which functions are: physical redirect, access to the network, error notification, frame distribution and flow control.

The transmitter takes the data of the Network layer and adds the layer header. Then the data is delivered to the physical layer which sends the frame to the receiver. In this device the flow of information is in the other direction, the physical layer gives the information to the Data link layer to be checked and delivered to the Network layer.

2.3 POLLING PROTOCOL

The traffic control can be done in different ways inside the Data Link layer. There are two kinds of classification; firstly, depending on the network typology, it can be centralized (a device, which is called “master”, controls the flow of information in the network) or distributed (the own devices of the network control the flow). Secondly, the traffic of information can be continuous (there is one device transmitting all the time) or burst (the devices transmit when they want). So there are the following options:

-If the network typology is centralized and continuous, one device controls the network and indicates when the other devices can transmit.

-If the network typology is distributed and continuous, no device controls the network. A token is used to indicate which device can transmit.

-If the network typology is distributed and burst, no device controls the network and there is not token. The devices are monitoring the network and when no device is transmitting, it can start to send the information.

The classification of the protocols is shown in the table 2.1.

	CENTRALIZED	**DISTRIBUTED**
CONTINUOUS	Polling	Token Bus Token Ring
BURST		CDMA/CD ALOHA

Table 2.1 Classification of the protocols

The polling is used in this thesis. The architecture of the network is master/slave. The master establishes connections with the slaves which are waiting to transmit until this moment.

One example of this network topology is shown in the fig 2.2 (the number of slaves can vary)

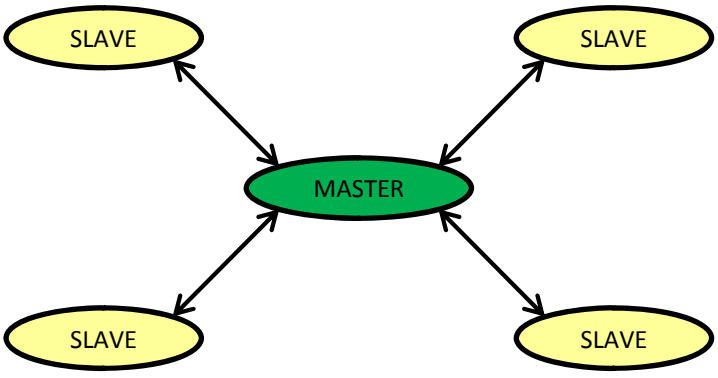

Fig 2.2 Network typology

The polling of the master can have several rules. The followings are used:

-The connection between master and slave is only uplink or downlink. It means, the slave cannot upload and download data in the same connection.

-The master makes always the same sequence of polling. Firstly, it makes an uplink connection with all the slaves and secondly a downlink connection in a cyclic manner.

-There is a maximum time per connection between master and slave. When it is reached, the master closes the connection with the slave, although the device has something more to send.

-The frame size is assumed to be 64 bytes.

These rules have been designed to minimize the design but they also reduce the bandwidth. It means that the design is easier and smaller but the rate of byte/seg transmitted is lower. The number of data frame transmitted per second can be increased with some improvements which are explained in the outlook part of this thesis (section 9).

The several protocols are not explained in this thesis but it is important to know the main pros and conts of the polling protocol. The advantages are:

-It has more reliability than the protocols with non-constant traffic. It is unlikely that two devices transmit at the same time therefore the number of collisions are less.

-In a master/slave typology, the master has the most complex hardware and software of the devices thus it is really expensive, on the other hand the slaves are simpler. In conclusion, the devices added to the network are cheaper.

-Error control. There exists a feedback which guarantees the reception of all the information. If some data has been lost in the way, this mechanism of control must order to resend the data.

While the disadvantages are:

-In this thesis is used a centralized typology, all the operations are made by the master. So if it fails, the network doesn´t work. While in the distributed typology, it is possible that one or more devices don´t work and the network still works.

-More reliability means more bandwidth lost. The lost time in opening-closing the connection is more than in the distributed typology.

-The network size is limited. When the master is designed, there are a maximum number of devices which can be connected at the same time. Therefore the size of the network is stipulated on the design.

When a kind of network and a protocol for the Data Link Layer must be chosen, it is important to study the main advantages and disadvantages and then try to find a balance between reliability and bandwidth.

2.4 MAC FRAME

The frame of the fig 2.3 has been defined inside the protocol. All the frames, which are sent inside the network, must have this format.

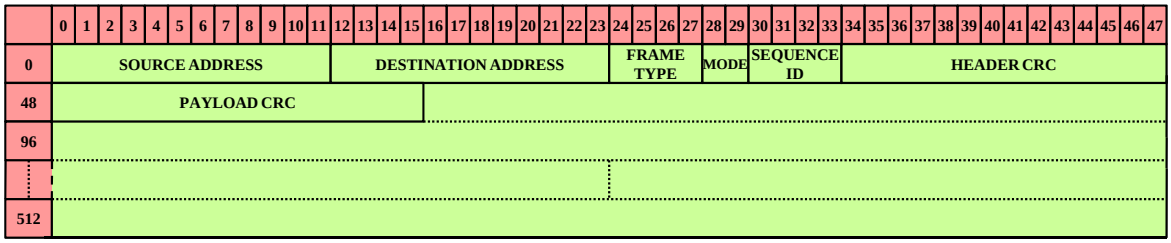

Fig 2.3 Frame format

First of all, there are two fields of twelve bits each one, the first is the address of the device which sends the packet and the second is the device direction where the transmitter wants to send the information.

After that, there is a field “frame type” of four bits which is the sort of the frame. It can be data or management. The type of the frame and the bits are shown in table 2.2:

BITS	TYPE
0000	POLL
0001	ACK
0010	NAK
0011	RTS
0100	CTS
0101	CNE + NAK
0110	CNE + ACK
0111	DATA
1110	TNE

Table 2.2 Type of the frame

POLL: It is used by the master to open a new connection with one slave.

ACK (Acknowledgement): It indicates the correct reception of a previous data packet.

NAK (Negative Acknowledgement): It is the contrary of ACK. It indicates the wrong reception of a previous data packet.

RTS (Request to send): It is used by the devices to indicate it has data for the other member of the connection

CTS (Clear to send): It is an answer packet of RTS. The device indicates that is ready to receive a data packet.

CNE-NAK (Coordinator node end plus NAK): The master indicates the connection is finished.

CNE-ACK (Coordinator node end plus ACK): The master indicates the connection has finished and the last data frame has been received correctly.

TNE (Terminal node end): The slave indicates the connection is finished.

DATA: It is used for a data frame, the receiver has to calculate the “Payload CRC” and check it. Otherwise these fields are ruled out.

The next field has two bits and it is used for the mode. There are two options; uplink connection (only the slave can send data packet) or downlink connection (only the master is allowed to send data packet). The table 2.3 shows the frame mode depending on the bits. The next field has four bits and it is the “Sequence ID” which helps to make the error control. Each frame is numbered thus if it has not the correct number, the receiver deletes it and returns to ask the same. All the fields including the latest make up the Header of the frame.

BITS	MODE
00	Uplink
01	Downlink

Table 2.3 Frame mode

After the header, there are two fields which are used by the receiver to check whether the content of the packet is correct. They are called “Header CRC” (14 bits) and “Payload CRC” (16 bits) that contain the CRC of the header and the data field. CRC is the acronym of cyclic redundancy check and it is explained in the section 2.5. The table 2.4 shows the polynomial generators used to create the two CRC.

CRC	GENERATOR POLYNOMIAL
Header	0x372B
Payload	0xBAAD

Table 2.4 CRC generator polynomial

The rest of the frame is data until 64 bytes. The fields “Payload CRC” and “Payload data” are the Payload of the frame and are used to send a data packet otherwise they are filled with zeros and the receiver doesn´t read it.

2.5 CRC

The cyclic redundancy check (CRC) or polynomial code checksum is used to detect errors in one block of data as the change of bits. It is frequently used in storage devices and digital networks. When a device uses this method, initially it calculates the CRC of a data block and then it sends the data block and the CRC obtained, other option is to store both together. Then if a different device wants to read the information of the data block, firstly it calculates the CRC and checks the result with the obtained earlier in the first device. If both are equals, the information is complete and correct, but the contrary means that one or more errors have occurred and the device should take some actions to correct it.

The mission of the CRC is to detect some errors. It is commonly used because it is very efficient. The calculation of the CRC is like a polynomial division where the remainder is the CRC.

The CRC helps to verify the integrity of data and if it is compared with the size of the data block, the number of bits added to the frame is not big. Meanwhile data security cannot be ensured, as for example the intentional alteration of data cannot be prevented by CRC/ARQ.

The fig 2.4 shows an example of how a CRC is obtained where is used the data block “11100110”. The generator polynomial has five bits that means the CRC obtained has four bits (the number of bits in the generator polynomial minus one). The result of this operation is “0110” which should be sent or stored with the data block and it will be necessary to check the integrity of the information.

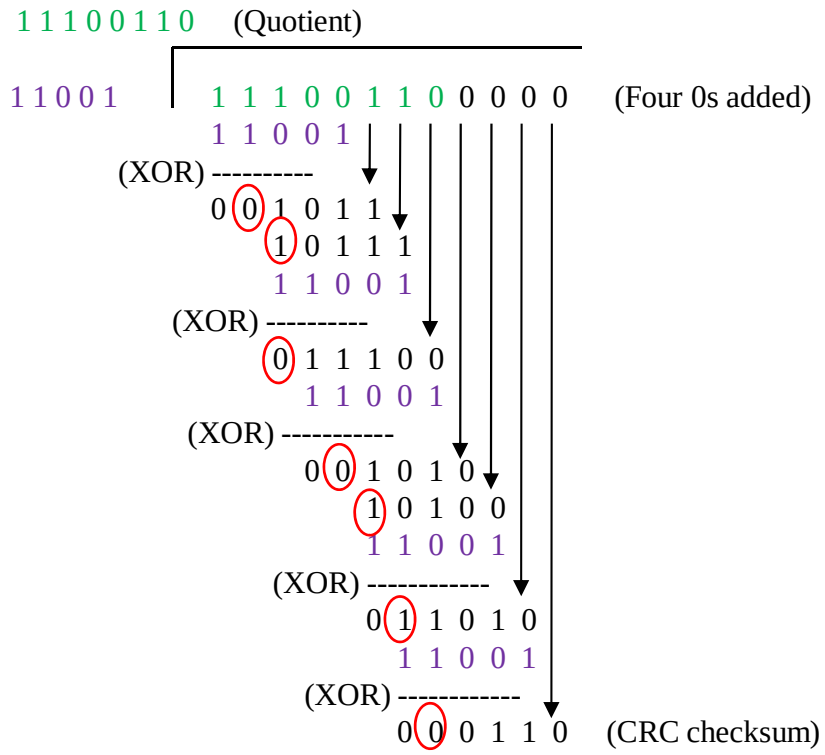

Fig 2.4 CRC example

The first step for the CRC calculation is to load the five MSB of the quotient (in this case) in one register. Then if the MSB of the register is '1', the XOR operation is made between the generator polynomial and this register. The result is stored in the same register. If the MSB is '0', the XOR is not made, so the register is shifted to the left and a new bit is loaded. The XOR operation only is made when the MSB of the register is '1'.

It is a loop until the entire quotient has been loaded. Then if the CRC has to be created, four zeros should be loaded (which is the number of bits inside the CRC minus one), while if it should be checked, the received CRC must be loaded. The MSB of the register is displayed with a red circle in the fig 2.4. When it is '0', a new bit is loaded while if it is '1', the XOR operation is made.

Another option is the parity bit which consists on look if the quantity of one's is odd or even. This method can detect only an odd number of errors. Other option is the parity bit in two dimensions but the possibility of error continues being very high.

Other alternative is the checksum where the block is divided into parts with the same number of bits and then they are added up. The addition is one's complement, if there is carry, it is added up in the least significant bit before it continues with the sum operation. This method is more commonly used in the Transport Layer.

3 BLOCK DESIGN

Before starting to implement the design in VHDL, it is very important to know what function has this project and to create a general design which is useful to divide the work into several parts. The first general design is shown in the fig 3.1.

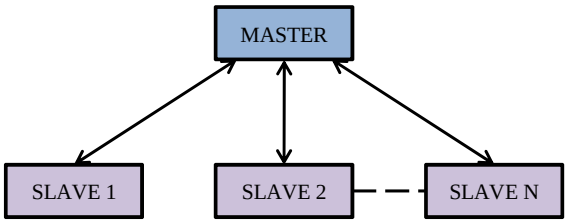

Fig 3.1 General design

There is a master which controls the flow of information inside the network and it polls the slaves. The number of slaves is limited by the design but this is not a design for manufacture, so this number depends on the slaves that each verification requires.

Once the general design has been made, the two fundamentals parts (master and slave) are divided into several sub-blocks. It means to make detailed design that is shown in the fig 3.2.

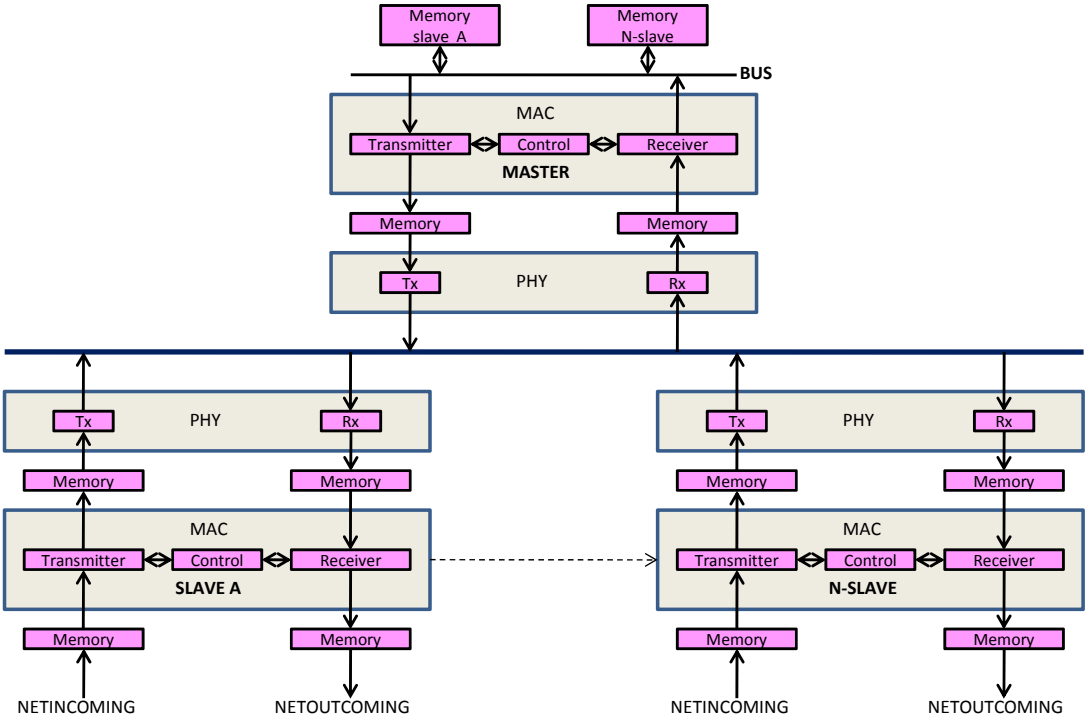

Fig 3.2 Block design

The master and the slave each have three sub-blocks inside: control, transmitter and receiver. There are several memories which connect the data link layer with the top layer and the bottom layer. Moreover the design has a physical layer because it is necessary to make the verifications. It is very simple and its mission is only to transfer data between memories but there is not codification of the data as in a real design.

The control block activates the receiver block when a frame has been received. Then the data is checked and the results are given to the control block which decides the answer that will be created by the transmitter block, once the control has given the information to send. All these frames are read or stored in the memories which interconnect with the other layers.

The operation of the three sub-blocks in the master is more or less the same as in the slave, the main differences are:

-The master control block has a state machine which controls all the slaves. It is working continuously, so it will never be placed in a waiting state unless it should wait to synchronize the network.

- The way of link the blocks with the memories is different in the master than in the slave. This is caused because the slaves contain top layers while the master has only Datalink layer and Physical layer for this design.

4 MASTER

The master controls the flow of information inside the network and it polls the slaves. It is very important because if it does not work well, the data transfer inside the network is impossible.

4.1 TRANSMITTER BLOCK

This part generates frames which are sent to the receiver. Everything is monitored by the control part which indicates when a frame must be created apart of the type and information to transfer.

There are two kinds of frames: management and data. The first is created easier because there is information only in the header and the payload is filled with zeros. While the second has information in the payload so it is necessary to extract the data of the memory, where they are waiting, and to calculate the CRC.

While the frame is being made, it is stored byte by byte in the output memory. When the process ends, the control indicates to the physical layer that the frame is ready and it can send it.

The architecture of this block is shown in the fig 4.1:

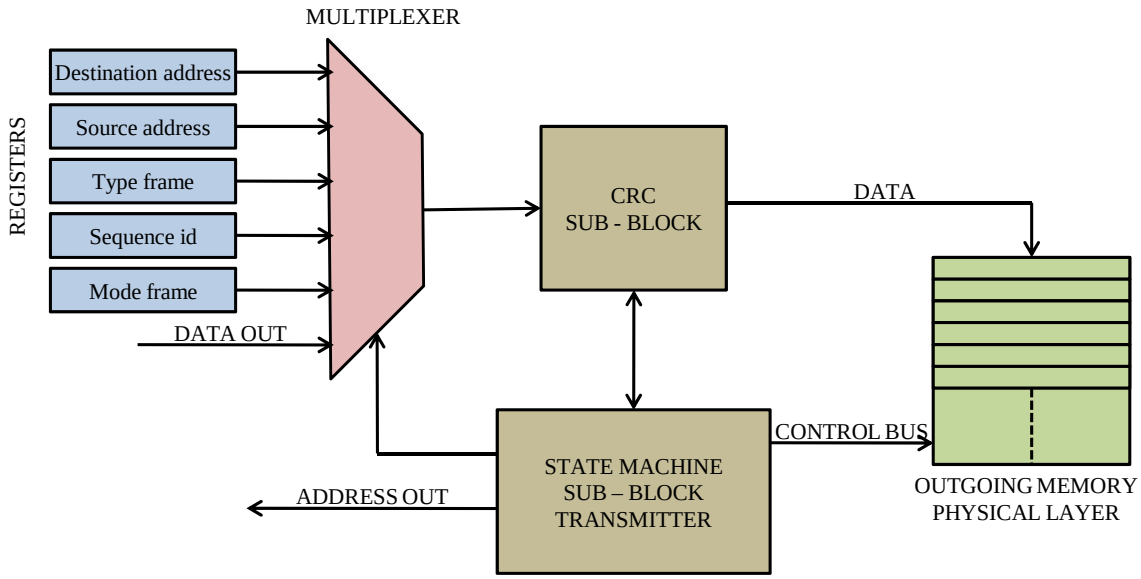

Fig 4.1 Architecture of the transmitter block

The simplified transmitter state diagram is shown in the fig 4.2:

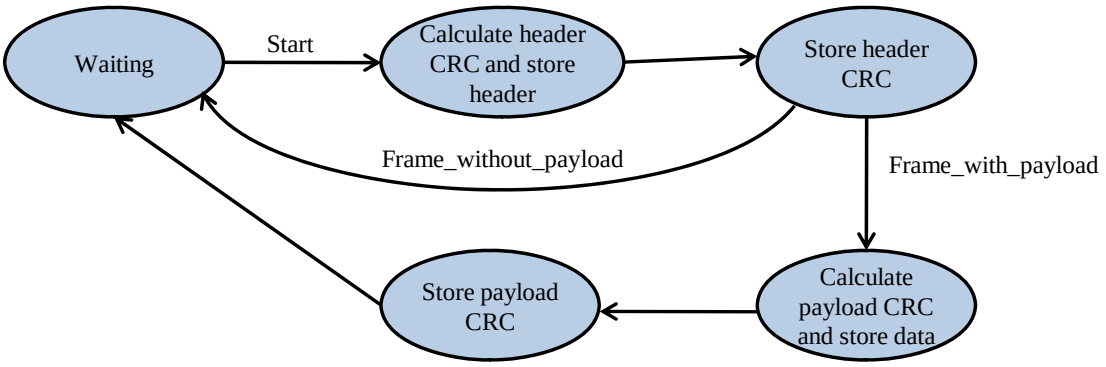

Fig 4.2 State diagram of the transmitter block

Initially it is placed in the waiting state until the control indicates that it must create one frame.

Firstly it calculates the header CRC. As is explained in the section 4.4, at the same time as the calculation is being made, the bytes are stored in the output memory which connects with the physical layer. For this task is necessary: the source and destination address, type of frame, mode of connection and sequence number. The control block saves this information in some registers. A multiplexer is used to choose these data.

When the CRC has been calculated, it is stored in the next 12 bits of the memory. In this point there exist two options, if the frame has payload or not. In the first alternative, the next step is to calculate the second CRC, in the other case the machine is placed in the waiting state where it indicates that the task is finished.

If the payload CRC must be calculated, it is important to leave empty two spaces of a byte because this information must be deposited after. Data used now is not obtained from the control block however it indicates the memory address where the data is placed.

When the second CRC has been calculated, it is stored in the two empty spaces that have been left with this aim. Then the machine goes to the waiting state where it shows to the control that its task has been accomplished and the physical layer can send the frame.

4.2 OTHER OPTIONS FOR THE TRANSMITTER BLOCK

Initially, it was thought the option in which first of all the two CRC are calculated, and then the frame is created. This is correct whether FIFO memories are used. There is no problem with the header CRC but the payload CRC should be placed before the data so it is necessary to wait until the calculation has finished.

The option chosen is to use conventional RAM memories then it is always possible to select the position for read or save data and it is not necessary to introduce them sequentially.

Theoretically this is the better way of make it because the relation between resources used and speed is favorable. The resources used are more or less the same quantity but it is faster because one cycle is saved per every byte stored, therefore 64 clock cycles are saved per one data frame. The complexity is in the design, it is important to know good the position inside the memory of each member because it is easy to put one byte in the wrong place.

4.3 RECEIVER BLOCK

This block analyzes the received frame and gives the results to the control. When a frame arrives, the physical layer decodes it, puts it in the memory, and then it indicates to the data link layer that the information is in the input memory.

At this point, the receiver has two important missions: firstly check the integrity of the data and confirm that all data have been received (both CRC are used to make this). Secondly, extract the information (source and destination address, type of frame, mode of connection and sequence number) and give it to the control block.

Like in the transmitter but in reverse, the data received must be placed in one memory. This step in the slave is different from in the master. In the first case, the memory connect the data link layer with the top layer while in the second case the memories only store the information.

There is one memory per each slave. When a frame is received, if the header is good, the control indicates to the receiver the memory and the position where it should store the payload.

The architecture of this block is shown in the fig 4.3:

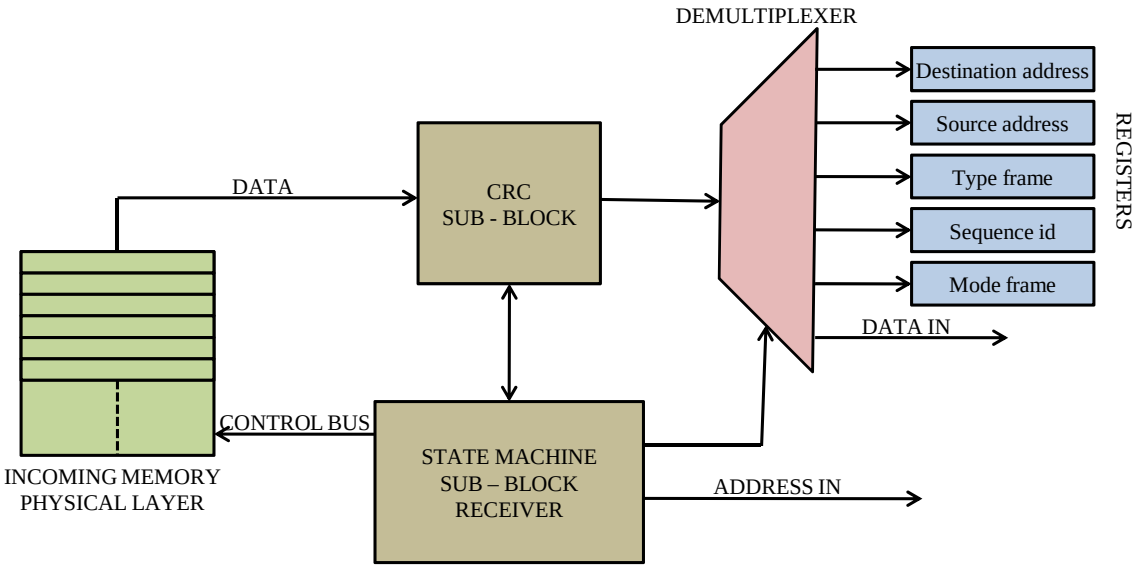

Fig 4.3 Architecture of the receiver block

The state diagram is shown in the fig 4.4:

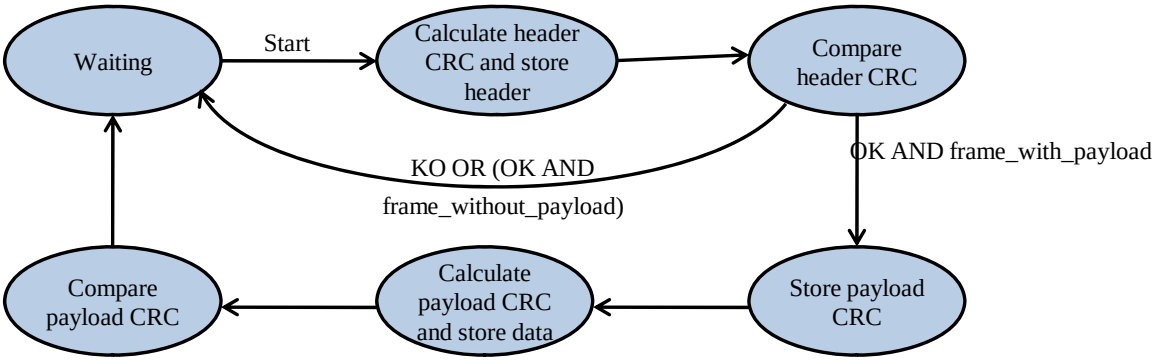

Fig 4.4 State diagram of the receiver block

At the beginning, it is on the waiting state until the control indicates that it can start to receive. The first step is to calculate the header CRC. At the same time it gives the information to the control with the aim to indicate the memory to use after.

When it has been calculated, the result should be compared with the received from the transmitter. If it is equal, it means all the information have been obtained correctly.

There are two options now: jump to the waiting state or calculate the payload CRC. It depends of: the header has been received correctly and whether the frame contains data. If both are true, the state machine goes to the state where the second CRC is calculated. While if one of this conditions is not true, the machine jumps to the waiting state.

If the payload CRC should be checked, the first step is to store the received CRC from the transmitter because the memory is read sequentially although it is not used until the end.

When the CRC has been calculated, it must be compared with the stored. If they are equal, it means the whole frame has been received correctly. The next step is to go to the waiting state to indicate to the control that the operation has finished and to remain here until the next reception.

To make the design of the memories, which store the data of each slave, there are several options. The most important are:

- At the same time that the payload CRC is being calculated, data is stored in an auxiliary memory. At the end of the calculation, if the result is correct, the information is transferred to the slave memory.

- At the same time that the calculation is being made, data is directly stored in the slave memory. The problem is to fill the memory with wrong information.

The option chosen is the second because the reception is faster. The machine has one state less, as result of the transfer between the auxiliary memory and the final memory is not necessary.

The main problem is to not introduce wrong data in the memory. A ring memory is created to solve this. Before fill the memory, the control indicates the first address position. If the calculation of the payload CRC ends correctly, this address is actualized in the control block (it means a 54 bytes increase). Otherwise a wrong result causes to not increase the address and the memory is overwritten after.

4.4 CALCULATION OF THE CRC

This block is included inside of the transmitter and receiver but it is important to show it and to explain how it works. The calculation inside of both is very similar. In the first is made with the aim of generate a new CRC while in the second should check the received CRC.

The diagram of the hardware used to calculate the two CRC is shown in the fig 4.5.

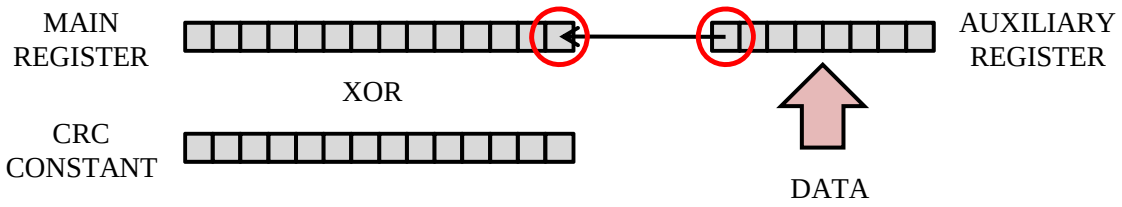

Fig 4.5 CRC architecture

The main blocks are the three registers and the input memory. First of all, a byte is extracted to the auxiliary register, it happens every time that it is empty and the process is not finished. Secondly, this register is moved to the left and the MSB is introduced in the LSB of the main register. Thirdly, the XOR is done when the MSB of the main register is '1', the result of this calculation is saved in the main register. The loop will end when all data have been extracted of the memory besides the zeros (when it is calculated in the transmitter) or the calculated CRC (when it is calculated in the receiver).

The state diagram, which shows the calculation of the CRC in the transmitter, is displayed in fig 4.6.

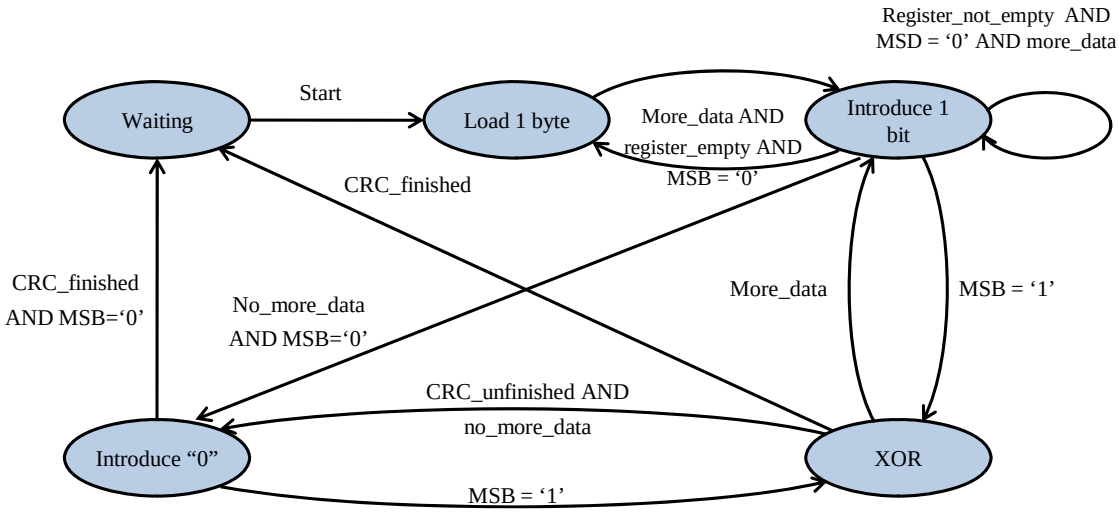

Fig 4.6 State diagram of the transmitter CRC

The initial position is the waiting state where all the counters and registers are reset. The first step is to load a byte in the auxiliary register. After this, the main register is loaded by a bit from the MSB (most significant bit) of the auxiliary register.

If the MSB of the main register is '1', the XOR between this register and the polynomial generator, which is set in the design, is done, otherwise when the MSB is '0', the XOR operation is not made and a new bit is loaded to the main register. This loop is made until the header or payload of the frame is full. In case there is not more data for send, the header payload is filled with zeros. Moreover when all the data bits have been loaded into the main

register (inclusive the zeros, whether they are necessary), some zeros should be loaded depending on the CRC size. For example, a CRC-16 (the size of this CRC is 16 bits) produces the adding of 15 bits.

When the loop has been finished, the machine jumps to the waiting state where indicates to the transmitter that the calculation has finished. There are two polynomial generators: one for the calculation of the header CRC and other for the payload CRC. They are following:

Header CRC constant (binary): 1 1 0 1 1 1 0 0 1 0 1 0 1 1 1
Header CRC constant (hex): 0x372B
Payload CRC constant (binary): 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 1
Payload CRC constant (hex): 0xBAAD

So the size of the header CRC is 14 bits while the size of the payload CRC is 16 bits. These have a bit less than the generator polynomial.

As is explained at the beginning of this section, the calculation of the CRC in the receiver is very similar as in the transmitter. The fig 4.7 shows the state diagram of the process in the receiver.

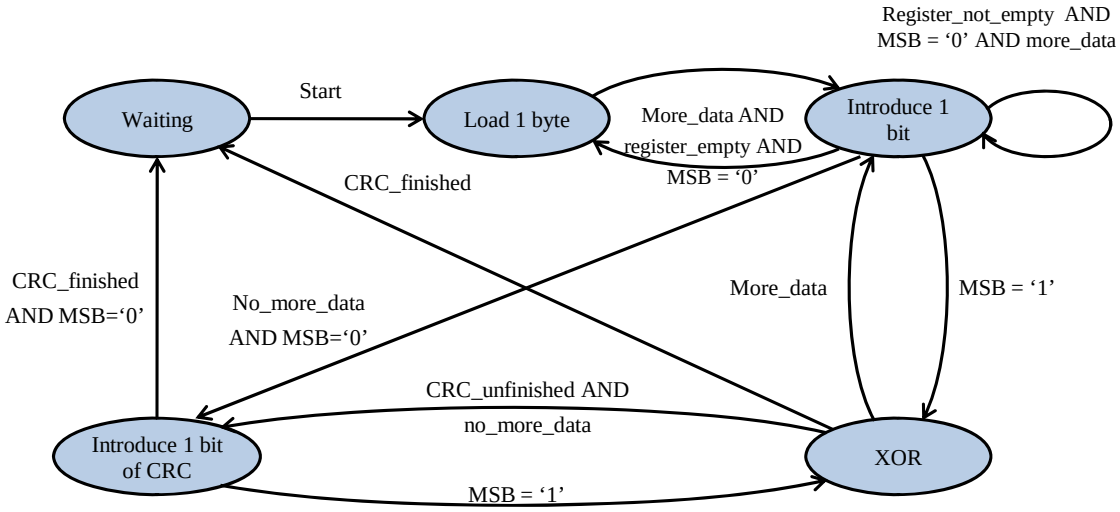

Fig 4.7 State diagram of the receiver CRC

The operation is the same as in the transmitter case but zeros are not added now at the end. Instead, the received CRC is added after the data. When the process has been finished and the data has been received correctly, the result of the main register must be all bits zero. Otherwise the information received is wrong so the transmitter should return to send it.

4.5 CONTROL BLOCK

This block controls the receiver which sends to it the information about the last frame received and then, the control decides the next step to make. Finally, it indicates to the transmitter the frame for send.

For make this block there are two options.

- The first alternative is to make three state machines: one for the uplink mode and another one for the downlink mode. The third state machine is a top state machine which controls both. This solution is easier but it is not optimum because some states are duplicated.

- The second alternative is to unify the two modes in one state machine. It is possible because some states in both modes are equals. The main problem is to know always the connection mode because the states are the same but the answer depends of the mode.

The option chosen is the second because the design is more optimal in time and resources. The architecture is shown in the fig 4.8:

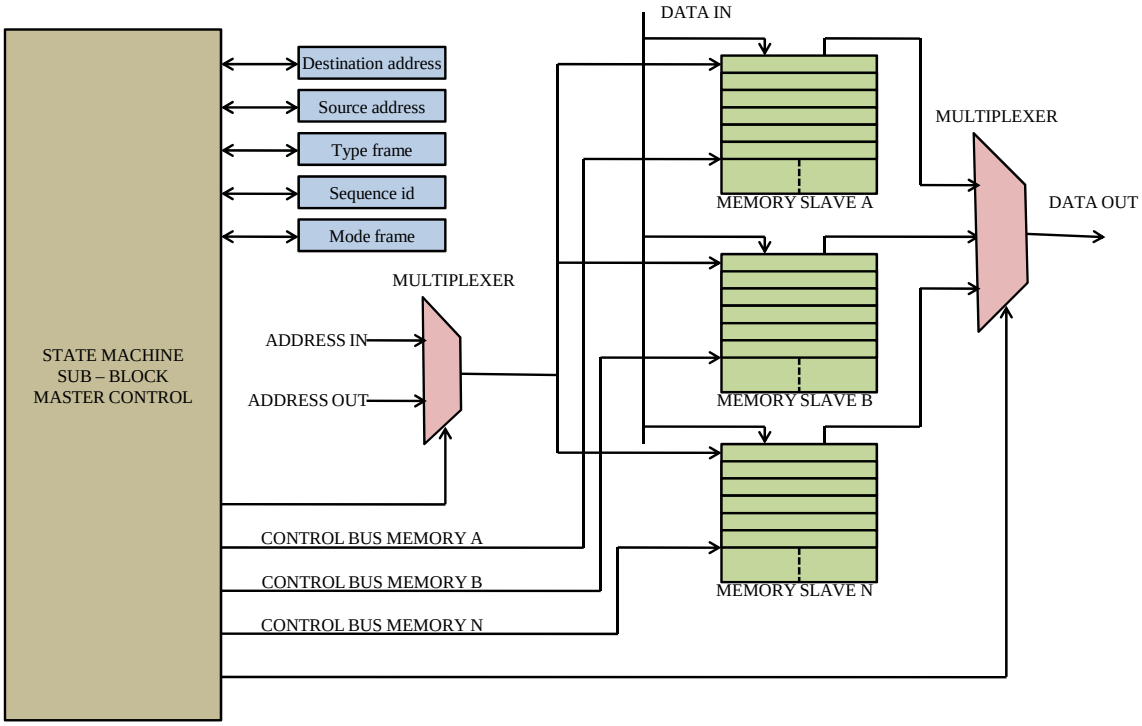

Fig 4.8 Architecture Master Control block

The main problem of the state machine is to jam in one point because the network stops to work. The critical points are the states which are waiting the reception of a frame. The possibilities of lost a frame or error frame are considerable so this is the reason of introduce some timers in these states and if the waiting time is very high, the machine jumps to other state which takes a corrective action to solve the error.

From this point, the explanation is divided in: opening the connection, uplink mode, downlink mode and closing the connection. In each section is shown its own part of the state machine and the whole state machine of this block is shown in the annex 10.3

4.5.1 OPENING OF THE CONNECTION

The initial state is “Decide mode and slave” where the master chooses slave and mode of connection. At the beginning of the design, two alternatives were studied.

-Upload and download data available in all connections. It means to be possible change the mode, once the connection has been established.

- Only possible download or upload data in one connection.

The second was chosen because the design is easier and the possibilities of error are lower.

The sequence of polling is always the same. There is no flow control mechanism. The master does not poll more often the slave which generates more traffic inside the network.

The sequence is to poll with uplink connections to all the slaves because initially, the memories inside the master are empty and there is not data available to download. When all the uplink connections have been made, then the master starts to make downlink connection. The part of the whole state machine, which is explained in this section, is shown in the fig 4.9.

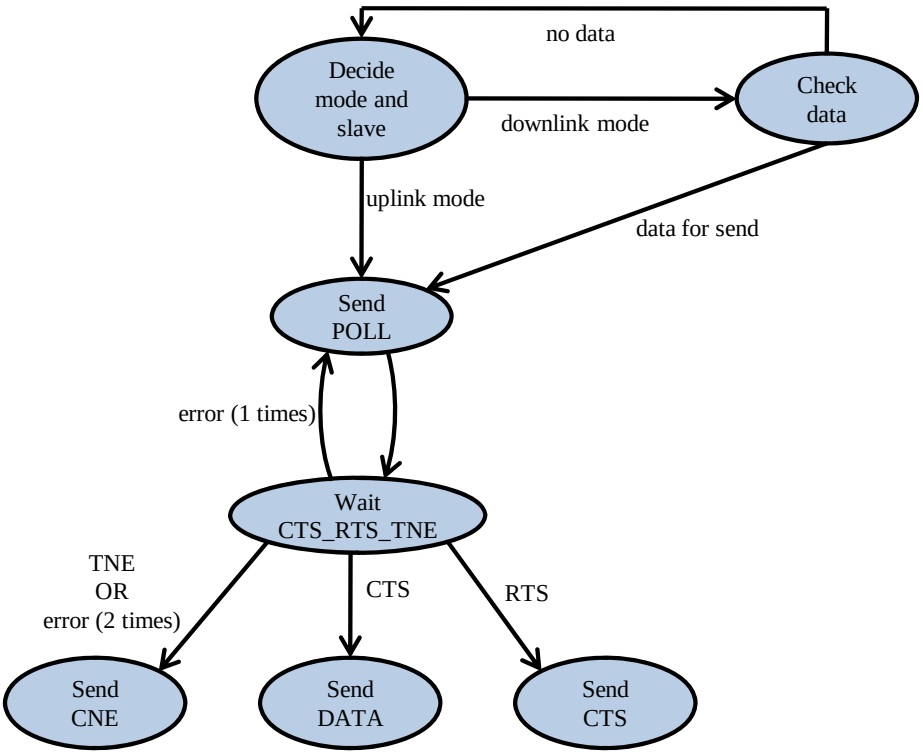

Fig 4.9 Opening the connection in the master

It does not have sense to establish a connection if the master does not have data to transmit. The solution is an additional state where the master, before to starting the poll, checks if there is data for the slave in an auxiliary memory, which is inside the master and is for the slave that has to be polled. If there is no data, the control checks the next slave.

All this is made by the states “Decide mode and slave” and “Check data”. The master jumps to the first every time that one connection has been finished and here changes the destination address and the mode if it is necessary.

If the connection mode is uplink, the master sends initially a “POLL” frame, however it is downlink then it goes to check if there is data. Two registers are used to make this operation. The first indicates until what address of the memory, the data has been stored. The second is the address of the last memory position read. If the number in the first register is higher than in the second register, it means that there is data to send.

In the “POLL” state the control sends the information (source and destination address, type of frame, sequence id and mode) that is used by the transmitter to generate the frame. When the frame has been sent, the control jumps to the “Wait_CNS_RTS_TNE” state

In this waiting state, it is important to keep several things in mind. If each mode had its own state machine, it would be only possible to receive a “CTS” frame or “RTS” frame in this point but in this design is possible to receive the two, therefore check the mode is fundamental. It is also necessary to check:

- The source address corresponds with the slave address.

- The destination address is the master address.

- The frame type is the correct.

- The sequence id is the following and the frame is not duplicated.

- The header CRC is exact and if there is data, the second is correct too.

If the frame received is correct, there are three options in this point:

-A “CTS” frame is received therefore the machine goes to “Send data” state where it starts to send data frames.

-A “RTS” frame is received so the master has to answer with a “CTS” frame and then the slave knows that it can start to send data frames.

-A “TNE” frame is received. This can happen for two reasons; first, in the uplink mode when the slave has not data to send and the connection should be closed. The other option is when the master is trying to open a connection with one slave but it does not receive well the frame. Then the “TNE” frame is used to indicate that the transfer is impossible and the master must try in the next round.

All these cases are under real conditions but the design should be made to solve some errors. The mechanisms of control used are the followings:

- A counter of error frames. It is incremented every time a wrong frame is received and it is reset every time a correct frame is received. If the number in the counter is two then the connection must be closed.

- The counter controls the wrong received frames but if it is not received, the machine is stopped in one state. The solution is to introduce some timers. When one of them reaches the maximum time, one counter is incremented. The second time that it happens, the connection must be closed. There are two different maximum timers, one is used when the control waits a frame with only header, and the other one is used when a complete frame is expected. Both are reset when a frame is received correctly.

If one of these errors happens once in the “Wait CTS_RTS_TNE” state, the “POLL” frame is sent other time. As it was created and stored in the output memory that connects the data link and the physical layer, the frame must not be created again. If these errors happen two times, the master has to close the connection with a “CNE_NAK” frame. It is not the same the “CNE_NAK” frame than the “CNE_ACK” frame, the explanation is in the section 4.5.2.

4.5.2 UPLINK MODE

The explanation of the state machine is divided in two parts from this point, depending on the connection mode.

If it is uplink and the slave has data to send, the master transmits a “CTS” frame which indicates the slave can start to upload data for the master. Then the machine jumps to the “Wait DATA” state. The part of the whole state machine, which is explained in this section, is shown in the fig 4.10.

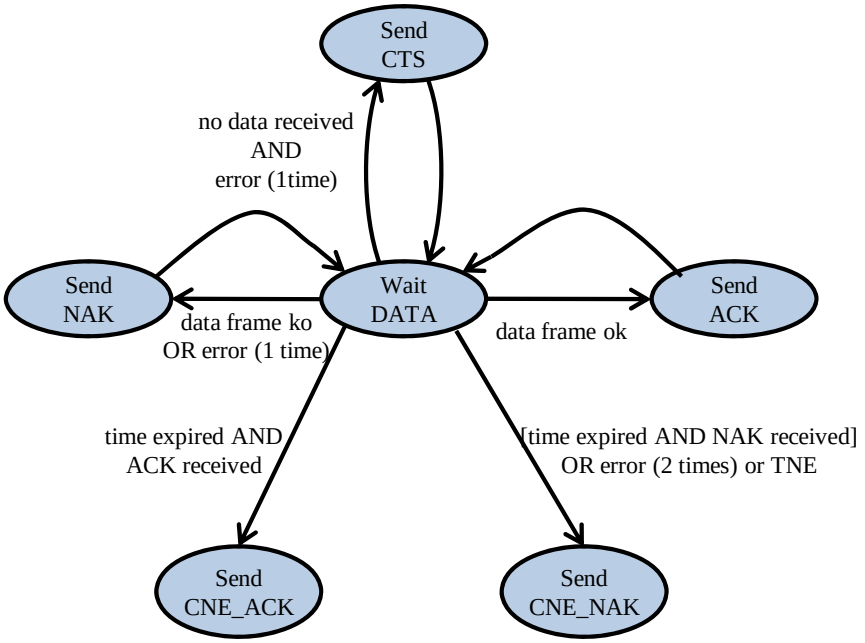

Fig 4.10 Uplink connection in the master

The receiver is waiting for a data frame in this point and it has to check both CRC; header and payload. But in this point, there are several alternatives apart of this case:

- A data frame is received correctly, it means both CRC, both address, type and mode of frame are correct.

- A TNE frame is received that indicates the slave does not have more information to send.

- A wrong frame is received because one or more fields have incorrect information.

- No frame has been received.

- The time per connection expires and then the master must close it.

In the first case, the machine jumps to the “Send ACK” state where a frame, which is called “acknowledgement”, is created. It indicates to the slave that its frame has been received correctly. The data is also stored in the memory that contains the information for the slave which the data is destined. So the register, which contains the address of the last written position, should be incremented. It is very important because if the update is not made, some data will be lost while if it is incremented wrong, some incorrect data will be transmitted. Once the “ACK” frame is sent, the machine returns to the earlier state and waits for receive a new data frame.

The second option occurs when the slave does not have more data to send and it asks the master to close the connection. The master answers with a “CNE_NAK” frame which is used for the closing.

The third happens when a wrong frame is received. It can occur in two different moments: just when a “CTS” frame has been sent, so the master sends another time the same frame because a corruption could have happened. The other possibility is that the master had received correct frames and it has a wrong frame now, the response is to send a “NAK” frame where the sequence id is very important. It helps for the slave to determinate the last correct frame received by the master.

The error count is incremented in this state too. If there are two errors consecutively, the master sends a “CNE_NAK” frame and closes the connection.

The fourth case is the same as the third. But the problem is to not receive the frame instead of receiving a wrong frame. The response is the same as in the earlier case and if it occurs two times, the connection is closed.

The last alternative is the time per slave expires. When the design was made, there existed two options: to close the connection after a certain number of frames have been sent or to close after a period of time.

The design uses the second. The time can vary a lot in the first option because if one slave has connection problems (as for example, it is very far from the master), maybe it has reception problems that could produce errors in the network and make it slower. This is the main reason of choose the time maximum per connection. It gives the chance to the slaves but if there are problems, the master does not spend much time with it.

When the time per connection expires, the machine waits until the next frame is received. If it is wrong, the answer is a “CNE_NAK” frame which means that the connection should be closed and the slave has to send the earlier data frame another time in the next connection.

While if the time expires and the last data frame received is correct, the master sends a “CNE_ACK” frame. Then the slave knows the good reception and it will not send again this information.

4.5.3 DOWNLINK MODE

If the connection has downlink mode, the machine jumps from the “Wait_CTS_RTS_TNE” state to “Send_DATA” state where the master creates a data frame with the information extracted from the memory which is for the slave connected. Then it goes to the “Wait_ACK_NAK” state. The part of the whole state machine, which is explained in this section, is shown in the fig 4.11.

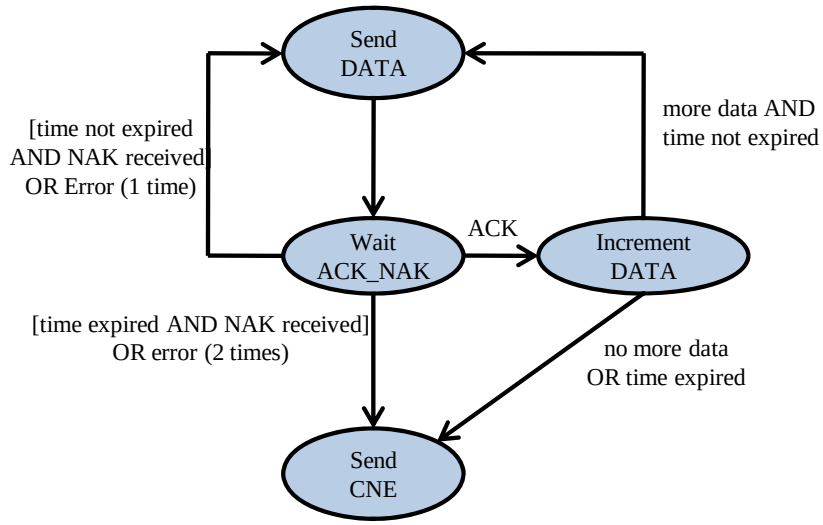

Fig 4.11 Downlink connection in the master

There are several cases in this point:

-A correct “ACK” frame is received. It means all fields of the frame should have the exact information.

-A correct “NAK” frame is received.

-A wrong frame is received. One or more fields have incorrect information or the CRC is not good.

-The maximum time per wait a frame is reached.

-The time per connection expires.

If a correct “ACK” frame is received, the machine jumps to the “Increment data” state which is explained after the five cases.

The response of the second and the third option is the same although the slave could have received correct the data frame and the problem could be in the transfer of the “ACK” frame. But it cannot be suppose because if it is not true, one data frame is lost. Then the answer of these issues is to send another time the earlier data frame or if it occurs two times, the master should close the connection with a “CNE_NAK” frame.

The fourth alternative can happen when the master waits during a period of time for a frame and it does not receive anything. If it occurs once, the master send another time the data frame while if it happens twice, the connection should be closed with a “CNE_NAK” frame.

When the maximum time per connection is reached, the master waits for the answer of the earlier data frame sent. If it is a “NAK” frame, the control jumps to the “CNE_NAK” state while with an “ACK” frame makes the same but before this, the machine goes to the “Increment data” state which is important for:

-The register which contains the last written address of the memory is incremented.

-After the register has been changed, it is checked if there is more data to send and more time before the maximum time per connection will be reached.

4.5.4 CLOSING OF THE CONNECTION

In the first design, the state machine went straight from the “CNE_NAK” state or “CNE_ACK” state to the state “Decide mode and slave”. This can cause some collisions in the network. The ideal cases are when the slave wants to close the connection and the “TNE” frame is received correctly, or when the master closes the connection with a “CNE” frame and it is received correctly by the slave. Meanwhile, the loss of these frames is very dangerous for the network because the states machines of both devices will not be synchronized.

The solution is placed an intermediate waiting state “Long wait” where the network is paused until the slave goes to its initial state and then the risk of collision is very low. The part of the whole state machine, which is explained in this section, is shown in the fig 4.12.

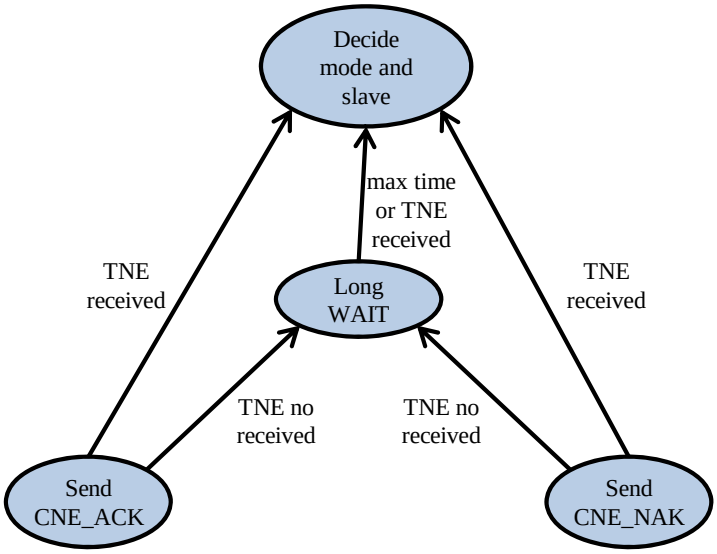

Fig 4.12 Closing the connection in the master

So after a “coordinator node end” is sent and if a “TNE” frame has been not received (it means that the slave has jumped to its initial state), the machine goes to the “LONG WAIT” state. The machine is placed in this point until a certain number of cycles have been reached or a “TNE” frame is received. Finally, the control machine goes to its initial state where it chooses a new slave.

5 SLAVE

This is the other fundamental part of the design. Like the master is divided into three blocks: transmitter, receiver and control.

5.1 TRANSMITTER BLOCK

The blocks of this design have been made as standard as possible. So the master transmitter can be used now, the architecture is shown in the fig 4.1

The main difference is when the master sends a data frame, it should choose the memory from where the information is extracted (depending on the slave that has to receive it). Meanwhile the slaves always extract the information from the memory which connects the data link layer with the network layer.

This issue is solved whether the task of choose the memory is given to the control block because the slave control cannot be the same as the master control. Then the transmitter block can be the same and it always extracts the information from the same bus where the master has to place the correct information.

5.2 RECEIVER BLOCK

As in the transmitter block, the slave receiver has the same structure as the master receiver shown in the fig 4.3 but the problem is in the storage of data now and not in the read of information. The solution is the same as in the other block. This task is given to the control block with the aim of make the receiver block standard.

5.3 CONTROL BLOCK

The state diagram of this block is different from the master control state diagram. The cause is because the master monitors and controls all the slaves of the network while the slaves are waiting until they are polled. The architecture is shown in the fig 5.1.

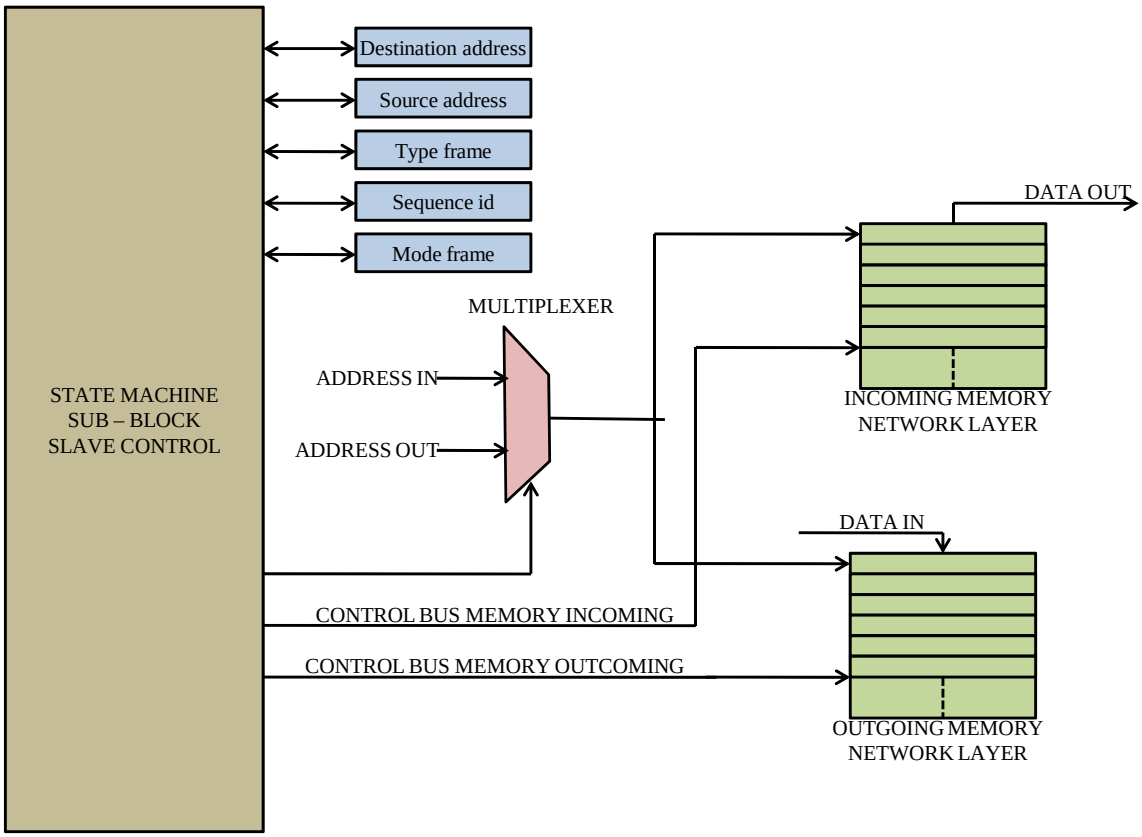

Fig 5.1 Architecture of the Slave Control block

From this point, the explanation is divided in: opening the connection, uplink mode, downlink mode and closing the connection. In each section is shown its own part of the state machine and the whole state machine of this block is shown in the annex 10.4.

5.3.1 OPENING OF THE CONNECTION

When the state machine is switched on, the first state is “WAITING”. The slave is reading all the network frames until one is destined for it. Meanwhile, if the frames are not correct, then the frame is discarded. The part of the whole state machine, which is explained in this section, is shown in the fig 5.2.

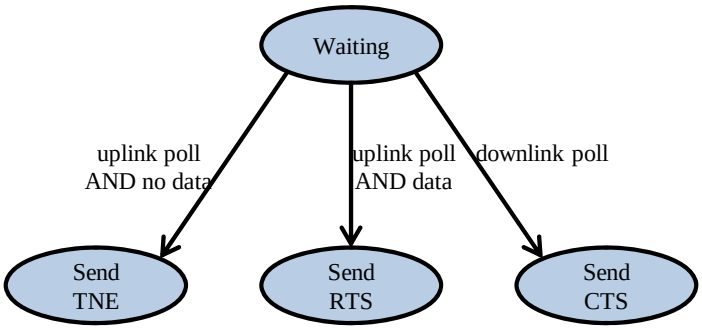

Fig 5.2 Opening the connection in the slave

If a correct management frame is received, the control has to check:

-The type of the frame is “POLL”. It means, the master wants to open a connection.

-The destination address should be the address of this slave because maybe the master wants to poll another slave.

If everything explained previously is true, the slave should read the mode of connection. There are the next options.

The first case is when the mode is downlink, the answer in ideal conditions is a “CTS” frame because it means that the master has something to send and the slave is ready to start the transfer of data.

The other option is when the master requires to upload data. But there are two cases inside of this:

-The slave has data to send, so it transmits a “RTS” frame which is used to ask permission for start the data transfer.

-The slave does not have data to send. The response to this case is to transmit a “TNE” frame with the aim to close the connection as soon as possible.

The slave control state machine is divided into two parts in this point, as in the master: uplink and downlink mode.

5.3.2 DOWNLINK MODE

After the slave has been polled by a frame with downlink mode and it has answered with a “CTS” frame, the machine goes to the “Wait DATA” state. The part of the whole state machine, which is explained in this section, is shown in the fig 5.3. Then four alternatives can happen.

-A data frame is received correctly.

-A frame is not received correctly, for example the frame is not full or one CRC does not coincide with the calculated earlier.

-The maximum time per wait a frame is reached.

-A “CNE” frame is received correctly. It means that the master does not have more data to send or the maximum time per connection has been reached.

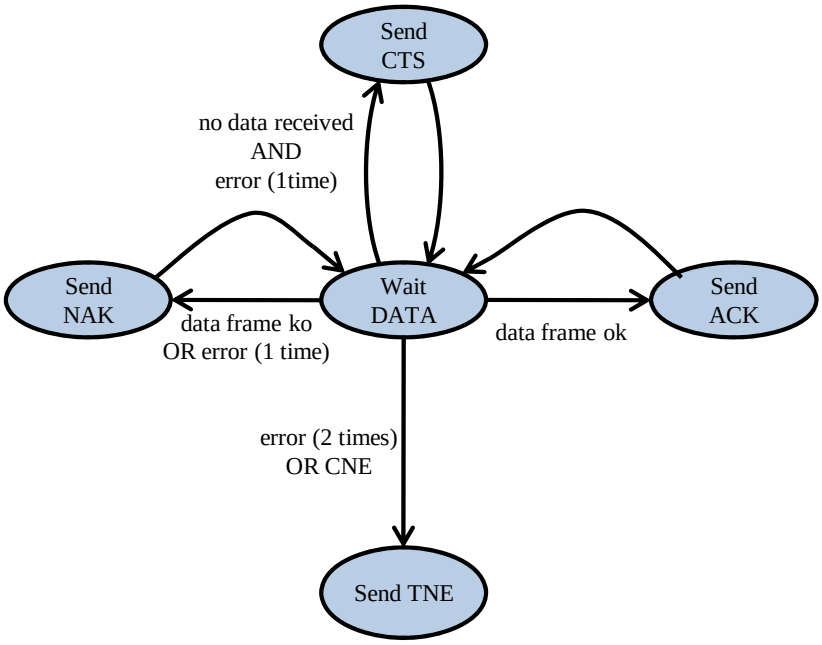

Fig 5.3 Downlink connection in the slave

The slave answers with an “ACK” frame in the first case and then it returns to the previous state for wait more data. The machine increases the register that contains the last address of the memory which has been written. It helps for the top layer to determinate how many data the memory has.

The second alternative is when some error happens in the creation, transfer or reception of the frame. The answer depends on the number of errors and whether some data frame has been received correctly during the current connection.

If the error occurs once and no data frame has been received, the answer is to send again a “CTS” frame. Because maybe the master has not received the earlier and it should know that the slave is ready for the transfer.

When the error occurs once and one data frame has been received, the slave send a “NAK” frame and the master should return to send the earlier data frame. Meanwhile if the error happens twice consecutively, the slave sends a “TNE” frame to close the connection.

The responses of the third case are the same as in the second. The main difference is that no frame is received in this case and the slave cannot wait more time.

The last case is when the connection must finish and then the slave answer should be a “TNE” frame.

5.3.3 UPLINK MODE

When the slave is polled by an uplink mode frame, the machine goes to the “RTS” state where this frame is created. Then it goes to the “Wait CTS” state where machine is waiting until happens one of the next cases:

-Some error occurs in the reception or the time maximum per wait a frame expires. The response is to send another time the “RTS” frame.

-The same as the earlier case but it occurs twice consecutively. The slave sends a “TNE” frame for close the connection.

-A “CTS” frame is received correctly. Then the machine goes to the “SEND DATA” state where it starts to create and send data frames.

When the frame is placed in the memory which connects the data link layer with the physical layer, the machine goes to “Wait ACK_NAK” state where can happen five cases:

-An “ACK” frame is received correctly

-An “NAK” frame is received correctly

-One error happens in the reception

-The maximum time per wait a frame is reached

-A “CNE” frame is received correctly

In the first case, the machine goes to the “Increment DATA” state where the register, which contains the last read address of the memory, is increased. After that, it is checked whether there is more data to send. In that case, the machine returns to the “Send DATA” state to send more data. But if there is no data, a “CNE” frame is sent to close the connection.

The next three cases produce the same response of the machine. If they happen one time, the earlier data frame is sent another time while if they happen twice consecutively, the slave sends a “TNE” frame to close the connection.

The response of the last case is always to send a “TNE” frame. But it is really important to know whether the last received frame has been a “CNE” plus “ACK” (which means the connection should be closed and the last data frame has been received correctly) or a “CNE” plus “NAK” (which means the same as the other, but one or more errors occurred in the reception of the data frame). If the earlier last data frame was received correctly then the register, which contains the last read address of the memory, has to be increased.

The part of the whole state machine, which is explained in this section, is shown in the fig 5.4.

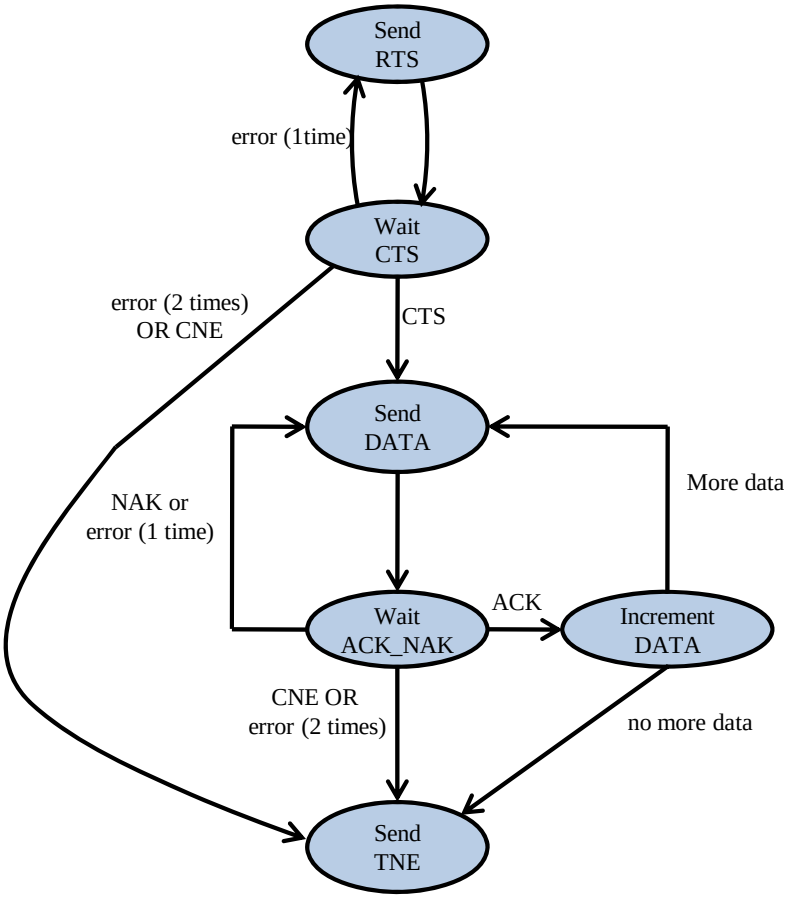

Fig 5.4 Uplink connection in the slave

5.3.4 CLOSING OF THE CONNECTION

Like in the master, theoretically the state machine should go to the initial state now. But if it goes directly the risk of collision is very high. The solution is placed a waiting state. The machine goes to this state only if a “CNE” frame has not been received because it means that the master is in the initial state and it has closed the connection.

The part of the whole state machine, which is explained in this section, is shown in the fig 5.5.

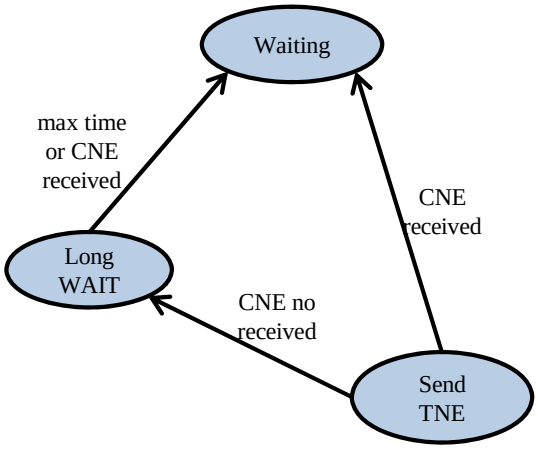

Fig 5.5 Closing the connection in the slave

6 VERIFICATIONS

The Project has been implemented in Modelsim. This program is a simulator too, so every change or improve of the code has been simulated. Its main advantage is that all the process, until the result is reached, can be studied.

The main problem is that although the simulations are very reliable, maybe it does not work in a real device. To solve this problem, the project is divided into two steps, which are tested in a FPGA, the ARQ scheme and the polling protocol.

The ARQ scheme is the first simulation. The aim of this step is to check the correct creation of the frames and the good reception of them. There are two devices in its architecture but no one works as master or slave.

The second simulation is made when the whole design has been made. It is necessary a physical layer to link the members of the network, so it is created. Now the devices work as master or slave and the architecture contains one master and three slaves. Finally, some data is introduced in the memories, as if this information comes from the network layer, and then the data transfer inside the network begins.

6.1 TOOLS

The design is written and simulated in Modelsim which is a program of the Mentor Graphics Company. The language used to implement the code is VHDL.

The design is synthesized by the program Quartus II. The place and route is made with it too. As result of this operation, a file with extension ".sof" is obtained. It is used to transfer all the design to the FPGA and then it can be tested.

Other important tool in the verifications is the "Cyclon III FPGA starter kit". As its name indicates, it contains a FPGA which is a Cyclon III model of the Altera Company. This kit is very useful because, apart of this device, it has some extras, e.g. memory blocks, switches, indicator.

The design uses the FPGA, the switches and the LEDs. The number of each depends of the verification that is made. The memory blocks are not used because the capacity of the FPGA is enough for the verifications made. Otherwise, if the design will be manufactured, it would be advisable to use this memory blocks. It is important that the capacity of the buffers is considerable because if the size is small, some data can be lost.

The rest important information about the components of the starter kit and the characteristics of the Cyclon III FPGA is placed in the annex 10.1.

6.2 ARQ SCHEME VERIFICATION

6.2.1 INTRODUCTION

The aim of this step is to test the correct creation of the frame and calculation of the CRC by the transmitter and to check the correct reception and calculation of the CRC by the receiver.

The transmitter creates frames with random data except the CRC field which is calculated. The other information is not important because the answer is only ACK (whether the frame has been received correctly) or NAK (a wrong frame has been received).

Moreover the frames have no payload. They only have headers, so the payload CRC is not necessary to calculate. In conclusion, there is no data transfer between the devices.

6.2.2 ARCHITECTURE

The architecture of this verification is very simple. There are two fundamental devices which do not work now as master or slave. They have receiver block and transmitter block but there is not a control block inside of them. All of this is placed inside the FPGA.

Also two button of the kit are used: one for reset the FPGA and then start from the initial point while the other is used to indicate when a frame has to be created and sent. The result of the simulation is shown in two LED.

The block diagram of this verification architecture is shown in the fig 6.1.

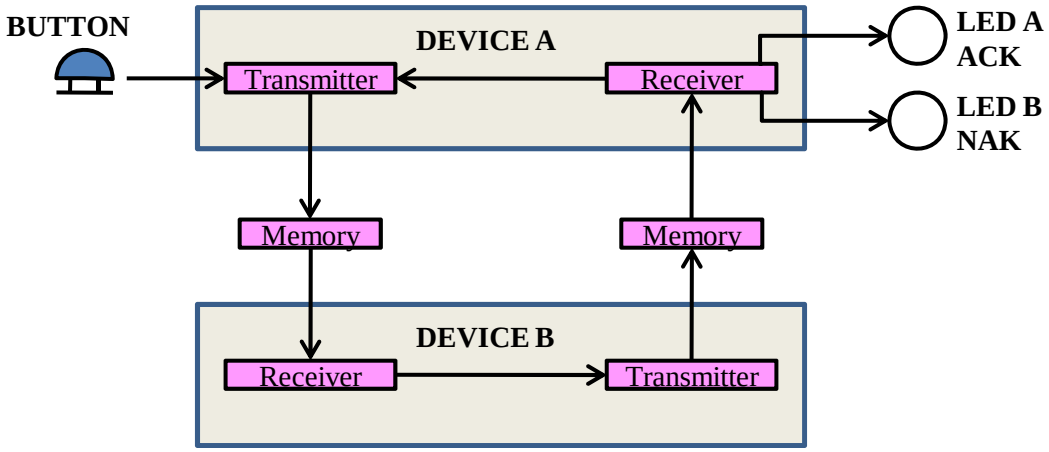

Fig 6.1 Architecture of the first verification

So when the second button is pushed, a frame is created by the device A and sent to the device B which has to check the entire frame and it responds with an ACK frame or NAK frame. The answer is received by the device A. Finally, this frame is tested and the results are shown in one LED. If the LED A turns on, it means an ACK frame has been received while the other indicates the reception of a NAK frame.

6.2.3 DEVELOPMENT OF THE VERIFICATION

It is very important to check the two possible results which are a wrong transmission and a successful transmission. So the first time that the button is pushed, a wrong frame is created. It has all the fields with random bits, so it is not correct because the CRC field is not calculated and the probabilities to be correct are almost zero (there is 1 probability out of 2^{14} because the CRC has 14 bits). The frame is shown in the fig 6.2.

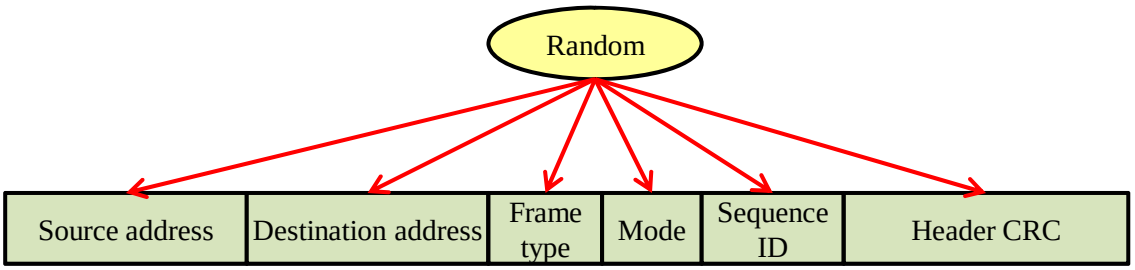

Fig 6.2 First frame generated in the first verification

At the end of all the process, there are three possible outcomes:

-The LED A turns on. It means that an ACK frame has been received.

-The LED B turns on which indicates the reception of a NAK frame. The causes of this can be the wrong creation of the frame by the device A or that there is a problem in the transfer from the device A to the device B.

-No LED turns on. It means that there is a problem in the network but it is very difficult to know where the error is produced. It can be in the creation, transfer or reception of the frame and in both sense.

If the simulation works well, the result should be the second solution. The others two indicates that there is a problem despite an ACK frame has been received because it is impossible in this case.

The second time that the button is pushed, a frame, which has all the fields except the CRC with random bits, is created. This frame is correct because the data inside the fields is not important. The receiver of the device B does not check the addresses, type and mode of connection or sequence id. It only checks the CRC that is calculated now. The frame is shown in the fig 6.3.

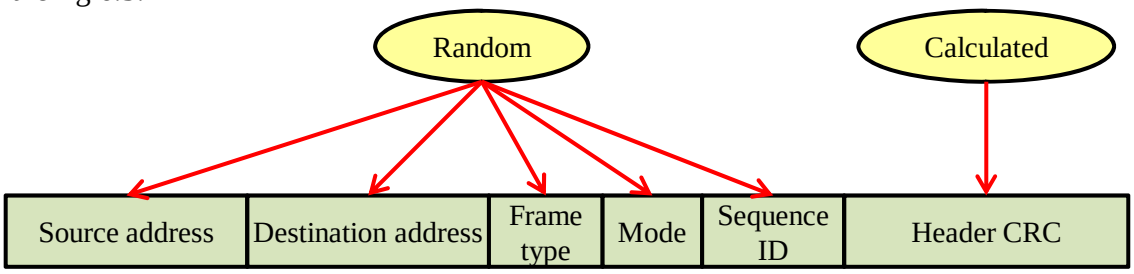

Fig 6.3 Second frame generated in the first verification

If the design works correctly, the result now should be the first possibility where a correct ACK frame is received by the device B and the LED A should switch on. In other case, if the LED B turns on, it means there is a problem in the frame transfer from the device A to the device B. While if no LED turns on, it means there is an error but the search of this is more difficult.

The fig 6.4 shows the process of frame transfer made between the both devices.

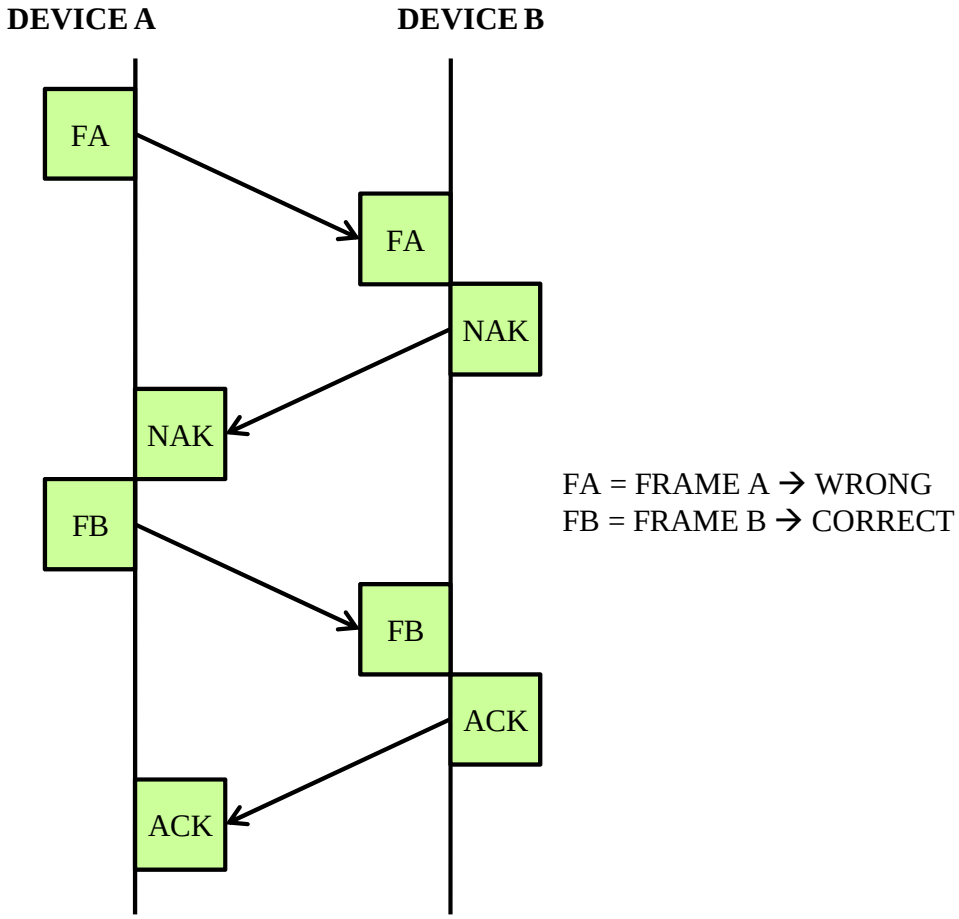

Fig 6.4 Frame transfer in the first verification

6.2.4 RESULTS

This verification works in Modelsim and in the FPGA. The result is correct, both LED turned on in the correct moment. The first time that the button is pushed, the LED B turns on. The second time, the LED A switches on.

Once the design works correctly, the next step is to continue with the rest of the design. The receiver block and the transmitter block has been made in this point, so the next stage is to make a control block which gives the order to these blocks. From this point, the devices are not the same because there are master and slave whose control blocks are different.

6.3 VERIFICATION OF THE WHOLE DESIGN

6.3.1 INTRODUCTION

This is the second verification that is made. The aim is to check the functional correctness of the master and the slave. A network is created to know the results, so it is necessary to create a simple physical layer which transfers the frames without codification.

Now there are a block inside the devices which controls the transmitter and receiver. The master control is different from the slave control. The connections are complete that means the two stages should be made: opening the connection, closing the connection and data transfer if it is necessary.

The frames have the two parts: header and payload. So there are data transfer inside the network, and the payload CRC should be calculated in the data frames.

To make the verification, some frames are placed in the slaves memories which connect the data link layer with the network layer. These frames are addressed to the other slaves, so at the end of all the data transfer process, it is checked whether the data has been placed in the correct position.

6.3.2 ARCHITECTURE

The architecture is a real network with one master and three slaves. The number of slaves can be incremented but this quantity is enough to check the design. It is important to know that all is implemented in the same FPGA. The physical layer is not ready now, so it is very laborious to create a network with the devices implemented in different FPGA, and it is important to remember that the mission of this project is to create a protocol for the data link layer and not to create a physical layer.

Despite of this, a simple physical layer is created to link the different devices placed in the network. It only transfers the frames but it does not codify them. When a frame is ready to send, the transmitter device indicates to the physical layer that the transfer should start. Then this frame is sending for all the incomings memories of all the devices.

It is made like that because the devices in a real network are reading all the network frames and if they are not for it, the frames are discarded. The other way of make this is transmit the frame directly to the device that has to receive it. It can be possible, but a real network would not be simulated.

The devices in the data link layer have all the blocks: receiver, transmitter and control. The control block tells the difference between the master and the slave because the receiver and transmitter are the same in all the connection members.

The fig 6.5 shows the architecture of the network but not complete (it is enough to understand the verification), it has only two slaves. The whole network with the three slaves is shown in the annex 10.2. This is like the fig 3.2, but the state of the memories is shown now.

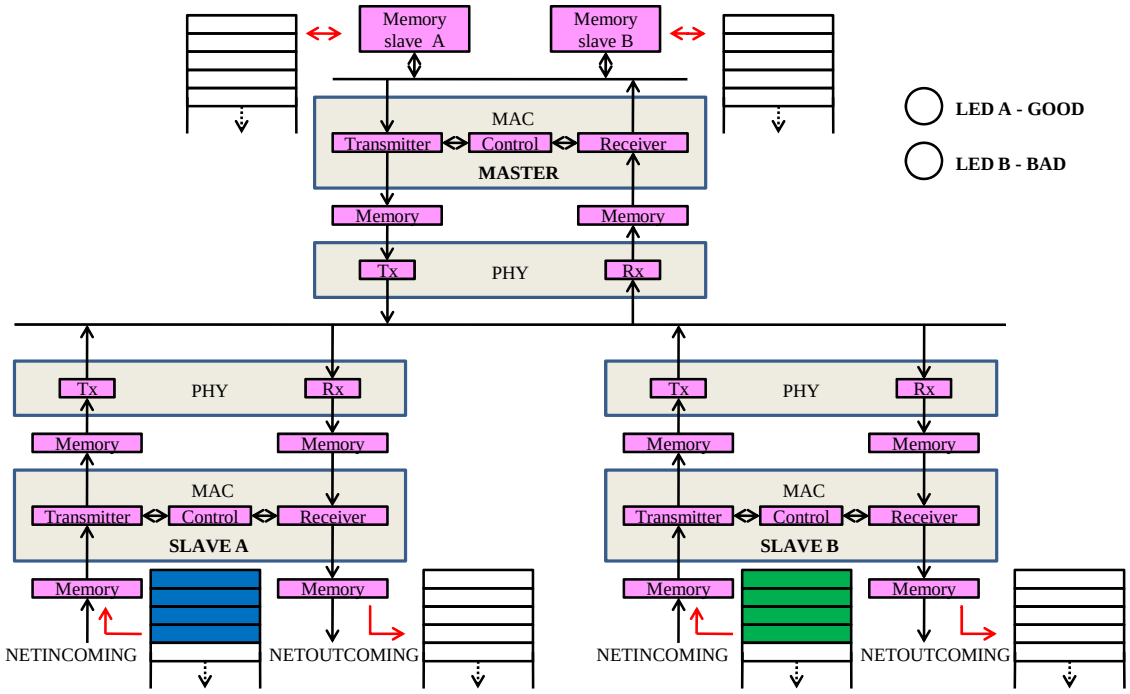

Fig 6.5 Architecture of the second verification

This fig 6.5 shows the initial state of the network, and as it is possible to see, there is data in the memories which connect the data link layer with the network layer. The blue data is for the slave B while the green is for the slave A

Apart of all this, two buttons and two LED of the kit are used. As in the first verification, one is used to reset the FPGA and place all the state machine in the initial state and the second is used to start the process. The two LED show the result of the verification. If the LED A turns on, the design works well while if the LED B switches on, it indicates there is an error in the network.

6.3.3 DEVELOPMENT OF THE VERIFICATION

When the button is pushed, the process is activated and the master starts to poll all the slaves with uplink connection. At the beginning, it does not have information to send, so it needs firstly receive some information for send after.

In this simulation every slave has data to send, so there is data transfer in all the connections. While the master is receiving the data frames, it stores them in the memory of the slave for which is directed.

At the end of this step, there is no data in the memories of the slaves because it has been uploaded to the master. So the master should deliver the data to each slave. The fig 6.6 shown the state of the network in this point of the process but the network implemented is shown in the annex 10.2.

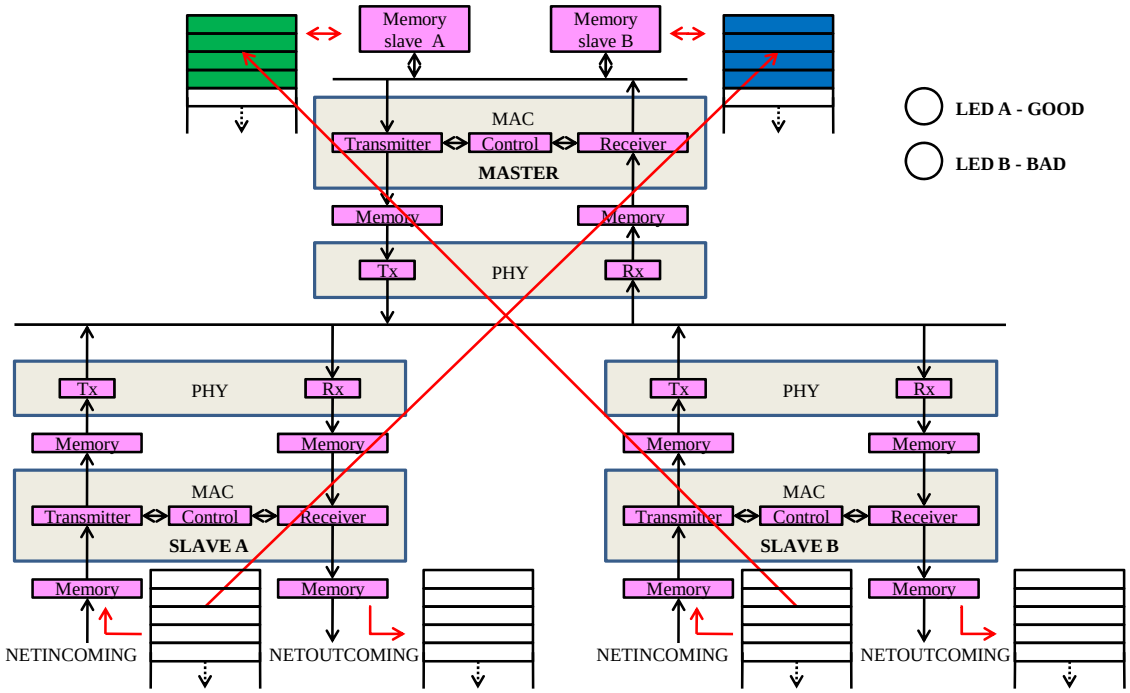

Fig 6.6 Transfer in the uplink connections

When the master finishes establishing uplink connections with all the slaves, then it starts to make downlink connections. It sends the data to each slave which stores it in the outcoming memories that connects the data link layer with the network layer.

At the end of this step, if the network works well and there is no problems in it, there is no data in the master memories and all the slaves have the data. The fig 6.7 shown the state of the network in this point of the process but the whole network implemented with the three slaves is shown in the annex 10.2.

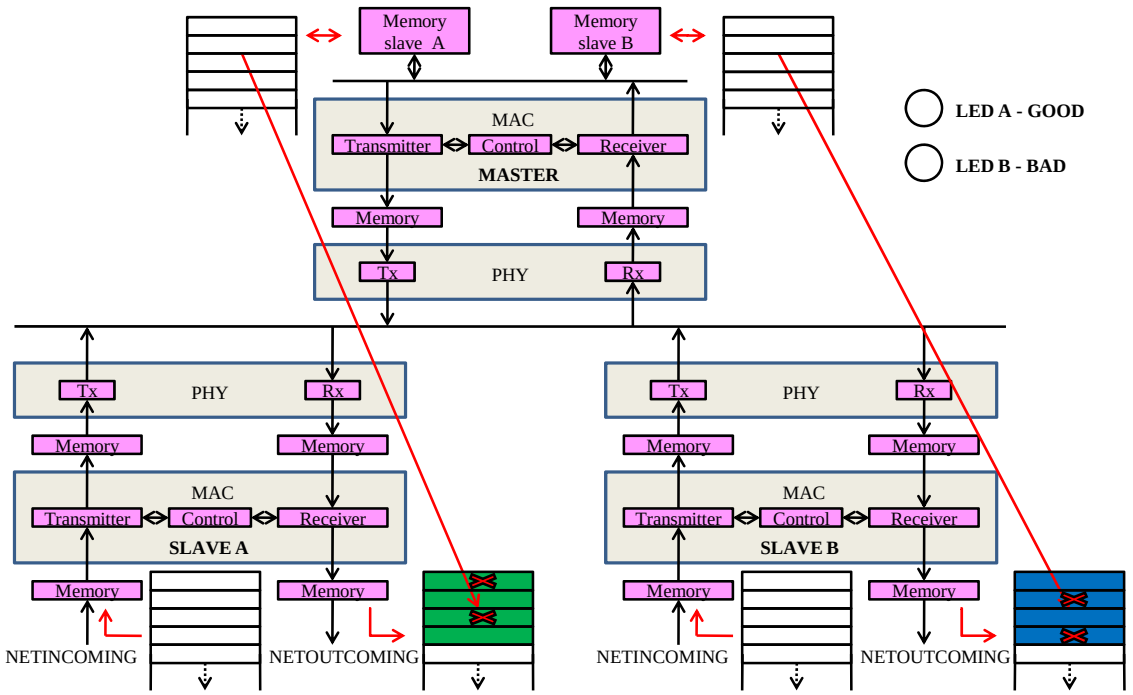

Fig 6.7 Transfer in the downlink connections

There is two ways of check the verification results, like in the first. The easiest is to simulate in Modelsim where all the process can be seen and it is useful to find the errors that can happen. Apart of the process, it shows the state of the memories at the beginning and in the end.

The second way, and the most important, is to implement in a FPGA. The main problem is to show the results because it is impossible to see the state of the memories. If the memory blocks of the kit are used maybe it would be possible to read the memories but the problem is the number of these blocks is limited.

The solution is creating a test block which is activated when the transfer process finishes. Then, this block reads some address of the slaves memories and makes a comparison between the bytes inside them and the correct bytes. If all the results of the comparisons are that both bytes are equal, the LED A is turned on which means the network and design works correctly.

If the LED B is switched on, some errors or problems in the network have occurred. It could be a big problem, if the simulation on Modelsim is correct because the entire network should be divided to check by parts.

6.3.4 RESULTS

This verification works in Modelsim and in the FPGA. All the process has been studied in Modelsim and when it was checked in the FPGA, the LED A turned on which indicates that all the data has been received correctly.

The resources of the FPGA used are shown in the table 6.1.

RESOURCE	RESOURCE USED	MAXIMUM RESOURCES	PERCENTAGE
Logic element	3287	24624	13 %
Memory block	114688	608256	19 %
Inputs/outputs	5	216	2 %

Table 6.1 Resources used in the second verification

7 IMPROVING THE CRC CALCULATION SPEED

Each transmitter block has inside a sub-block which calculates the new CRC and each receiver block has a sub-block which checks the received CRC. Both sub-blocks have been made by creating a new VHDL but another possibility is to use the mega function.

The main problem is that the code used is twice as slow (in the worst case) than the mega function. As is shown in the fig 4.6 and fig 4.7, the CRC is calculated in two steps: firstly is loaded one bit to the main register, and secondly, if it is necessary, the XOR operation is made. Theoretically these two steps can be unified and made in only one step. In fact, this is the mega function algorithm.

Moreover, there are more methods to do it faster. The main problem is the quantity resources used for this, but once the free resources inside the FPGA are known, if there is place, these methods can be used.

One method is to take advantage of that the polynomial generators are always the same. Then it is possible to create a table with the results, and then in one step can be loaded more than one bit. The architecture is shown in the fig 7.1

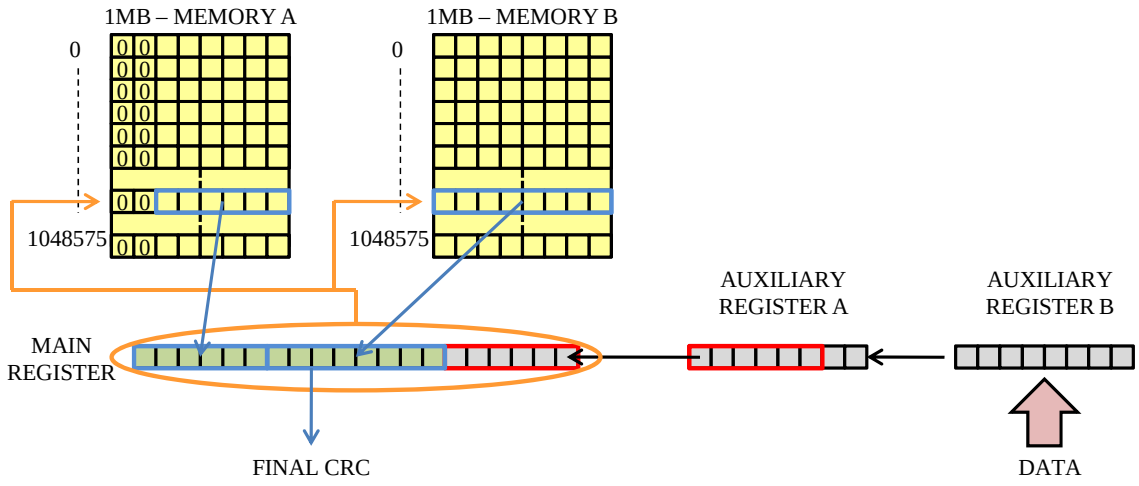

Fig 7.1 Architectur of the CRC with the improvements

The example has been created for the header CRC which has 14 bits. So if two memories of 1 MB are used, it means that the address busses are formed by 20 bits. Then instead of the load 15 bits and the XOR operation, the main register is the address bus of the memory and the output from the memory is the result.

Theoretically, the result has 16 bits (8 bits per each memory) but the two MSB are zero. So the rest of the bits are loaded in the MSB of the main register, while the six LSB of the main register are loaded with new bits.

Moreover there are two auxiliary registers. Sometimes in the charge of the six bits, maybe there are less bits in the register, so it is need a load of a new byte in the auxiliary register and then the load of the main register should be completed. These two registers avoid spending two more cycles.

This architecture uses less clock cycles for calculating the CRC but this number depends on the speed at which data is extracted from the memory. And it can only be used, if there are free resources in the FPGA at the end of all the design (inclusive the other layers).

8 CONCLUSION

Once the design has been finished, the conclusions are it works in Modelsim and in the FPGA. It is different because in Modelsim, all the connections can be studied and all the process can be seen with detail. Meanwhile the FPGA simulation is more important but only the solution can be seen and not what happens until this point is reached.

Some error cases could not be tested although the design is done to solve them but the fact that it could not have been checked in a real hardware and real physical layer, it makes impossible to test what happens in the reality.

The speed of calculation the CRC can be increased. Proprietary code is implemented in this project instead of using the core generator to create mega functions. So the calculation of the CRC is twice as slow (in the worst case).

The design has been made with the mission of obtain an optimum relation between resources used and speed, as the design is not made for an industrial use. In other conditions, maybe it would be more important to use more resources and have more speed, or the contrary. So a midpoint in the both characteristics has been tried to reach.

9 OUTLOOK

The fact that the design could not yet be tested with all the members of the network implemented in separate FPGA and connected by a real physical layer, it makes impossible to know the real performance of this design.

There are several possible improvements for this design, but the necessity of them depends on the physical layer performance: The improvements can be:

-Make the CRC calculation faster (section 7)

-Variable size of the data frame

-Download and upload available in every connections

-Control the error frequency per slave

-Variability of the time per connection

9.1 VARIABLE SIZE OF THE DATA FRAME

The data frame size is always the same. The main problem is sometimes there is no data to fill all the payload of the frame, so it is completed with zeros. It is not the best solution because there exists a lost time in sending and calculating the CRC of some data that after is discarded. There are three solutions.

The first is to not make the size of the data frame variable but to introduce a new field in the header which contains the number of data bits that are in the payload. There is lost time in sending no valid data (because the frame should be filled with zero too) but the lost time in calculating the CRC is avoided now.

The second solution is to make the same as in the first but the size of the frame can vary now. The two lost times are avoided now.

The third solution is to make variable the size of the payload frame but without adding a new field. The main problem is in the receiver because it does not know how many bytes the frame has and therefore the payload size. The physical layer should help to solve this problem, because it knows how many bytes it has stored in the memory that connects this layer with the data link layer.

The best solutions are the last two because they avoid the two lost times explained earlier. The problem of the second is the architecture of the frame should be changed but it is more secure. While in the third solution, the physical layer provides the size of the frame, so the number of bytes used to calculate the payload CRC should be an input instead of a constant. The problem is that the physical layer should work really well because the number of errors inside the network can be increased.

9.2 DOWNLOAD AND UPLOAD AVAILABLE IN EVERY CONNECTION

In the design, the master sequence is always the same. It establishes uplink connection with all the slaves, and then downlink connections, so it is a loop. One problem of the polling protocol is the lost time in opening and closing the connection. Moreover this time is greater, if it is not allowed to upload and download data in every connection.

The solution is to make possible the change of connection mode, once it has started. The state machines, inside the control block of the receiver and transmitter, should be changed and two new types of frame have to be created. One should be used by the master to indicate that wants to change the connection mode and the other must be used by the slave to answer.

The main problem of this method is the synchronization of the two devices involved in the connection. The two members have to change the mode of connection otherwise some incorrect frames are introduced in the network.

9.3 CONTROL THE ERROR FREQUENCY PER SLAVE

The errors in the network can be produced for so many reasons. Sometimes, there are slaves with which are impossible to open a connection, so it is not optimum in time try to make this in every loop of the master sequence.

The solution is to introduce two counters per slave inside the master. The first counter indicates the consecutives times that it has been impossible to open the connection with this slave. The second indicates the consecutives times that the master has not tried to open the connection with this slave.

Then these counters are read by the master when it should open a new connection with the slave and for example, if the slave has failed thrice consecutively, maybe the best solution is to wait one or more loops of the master sequence for trying to open a new connection.

9.4 VARIABILITY OF THE TIME CONNECTION

There is a maximum time per connection in this design. The main problem is when a slave (or it can happen in the master too) has a lot data to send and in every connection the maximum time is reached, so each time, the memory is more full. It is very dangerous because if the memories are full, there will be data loss.

The solution is to introduce a counter per slave in the master which indicates the consecutives times that the maximum time per connection has been reached by this slave. Then the master can increment the time per connection, if this number is very high.

10 ANNEX

10.1 CYCLONE III FPGA STARTER BOARD

The fig 10.1 shown the board used to make the both simulations. The blocks used in the verifications are the Cyclone III FPGA, the LEDs and the switches.

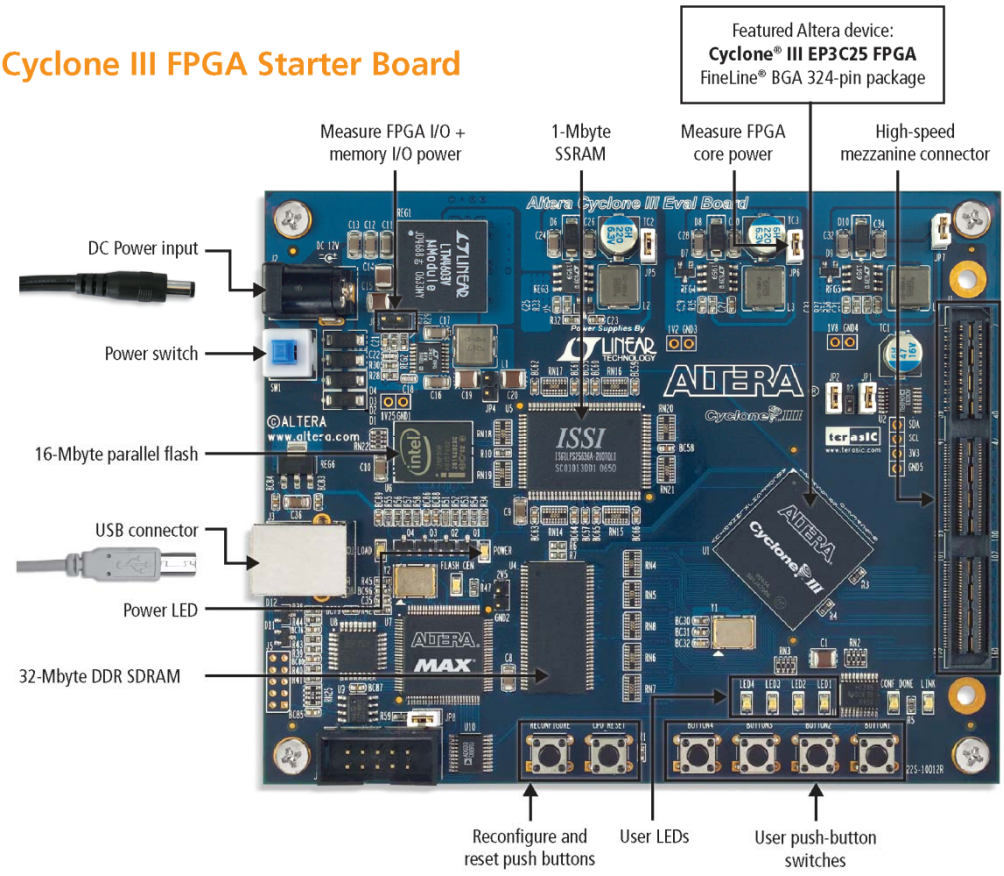

Fig 10.1 Cyclone III FPGA Starter Kit

10.2 SECOND VERIFICATIONS

The figures shown in the section 6.3.3 explain the development of the second verification but the architecture in these figures is not complete. There are two slaves and the verification was made with three slaves. The followings figures show the real process of this verification

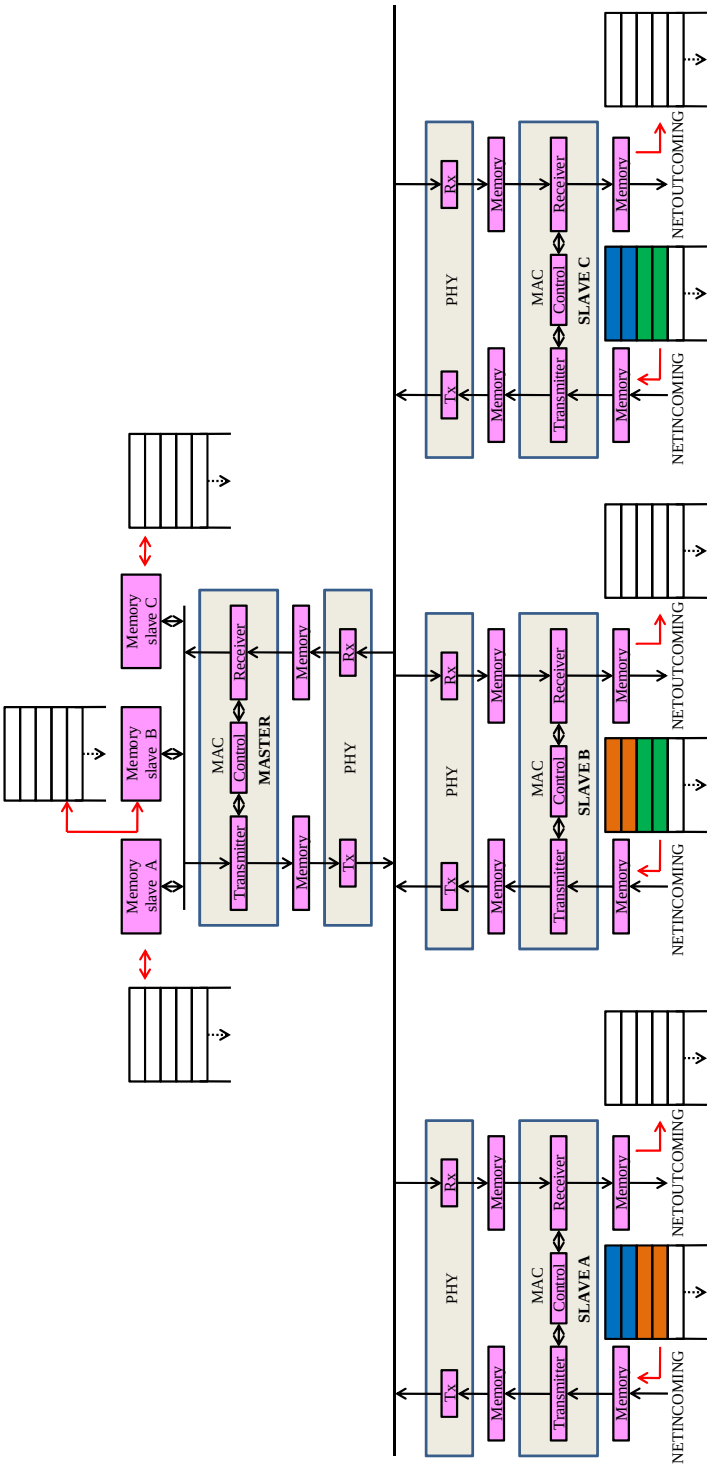

Fig 10.2 Architecture of the complete second verification

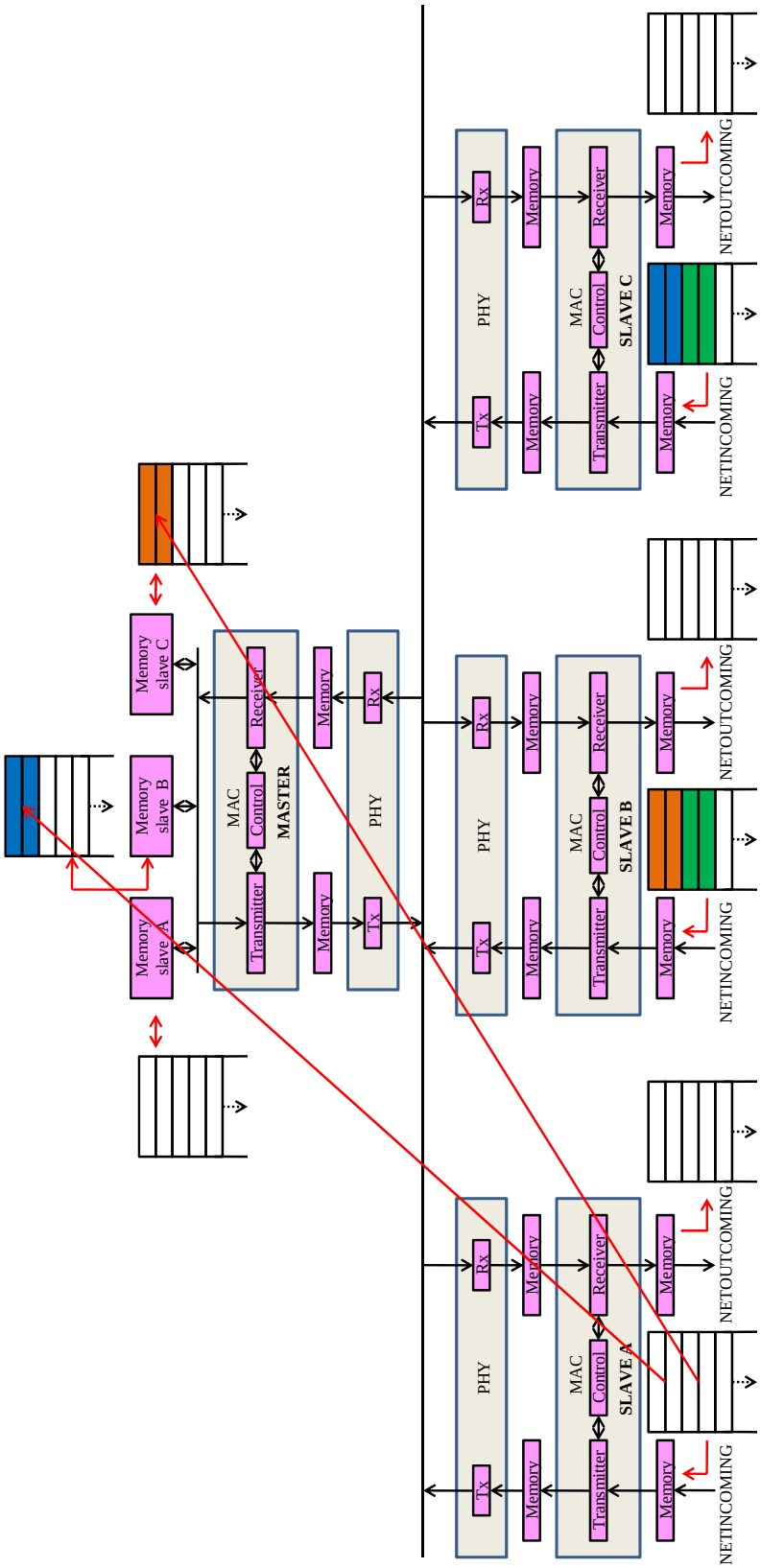

Fig 10.3 Second verification: Master polls Slave A with uplink connection

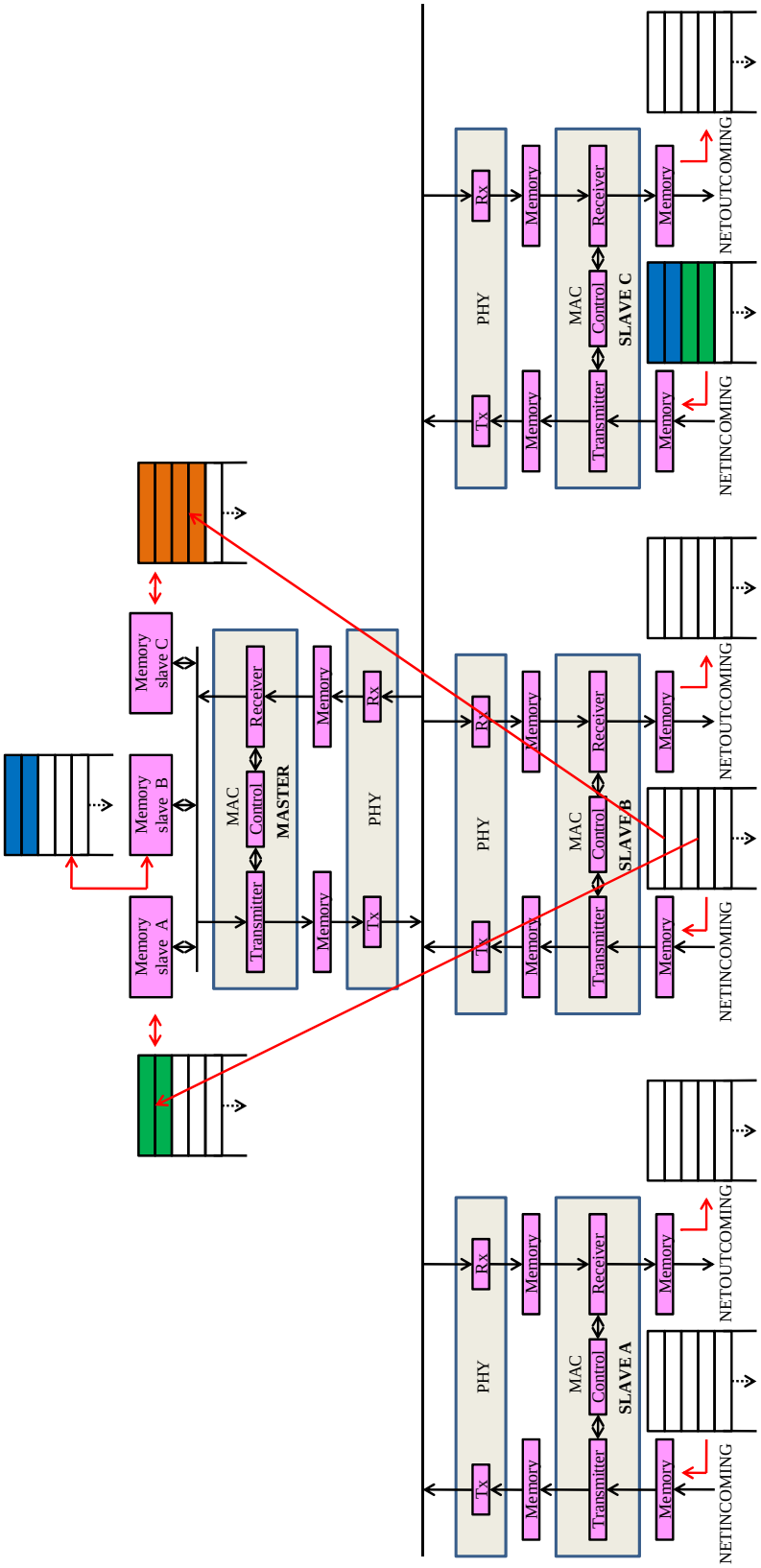

Fig 10.4 Second verification: Master polls Slave B with uplink connection

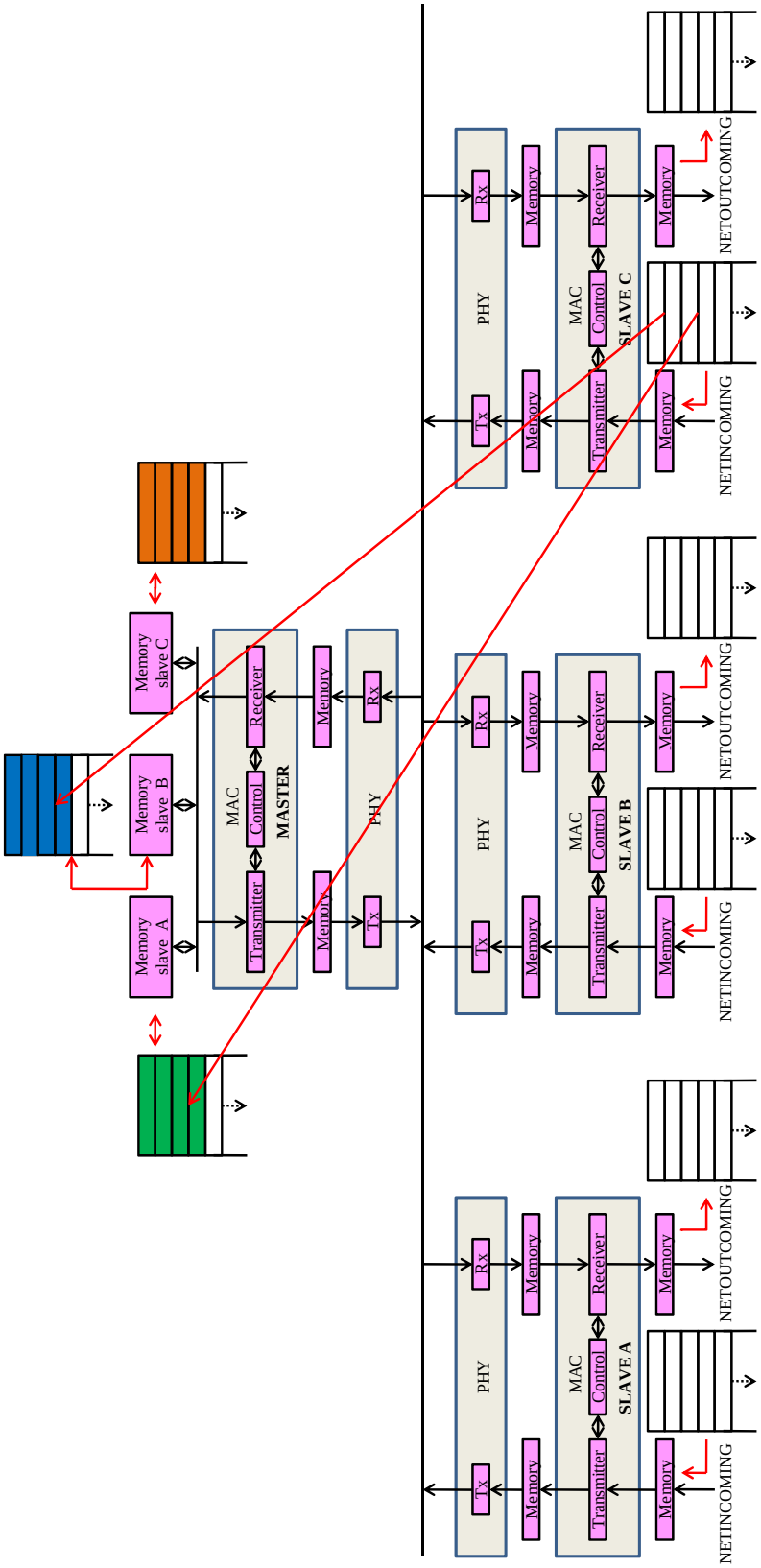

Fig 10.5 Second verification: Master polls Slave C with uplink connection

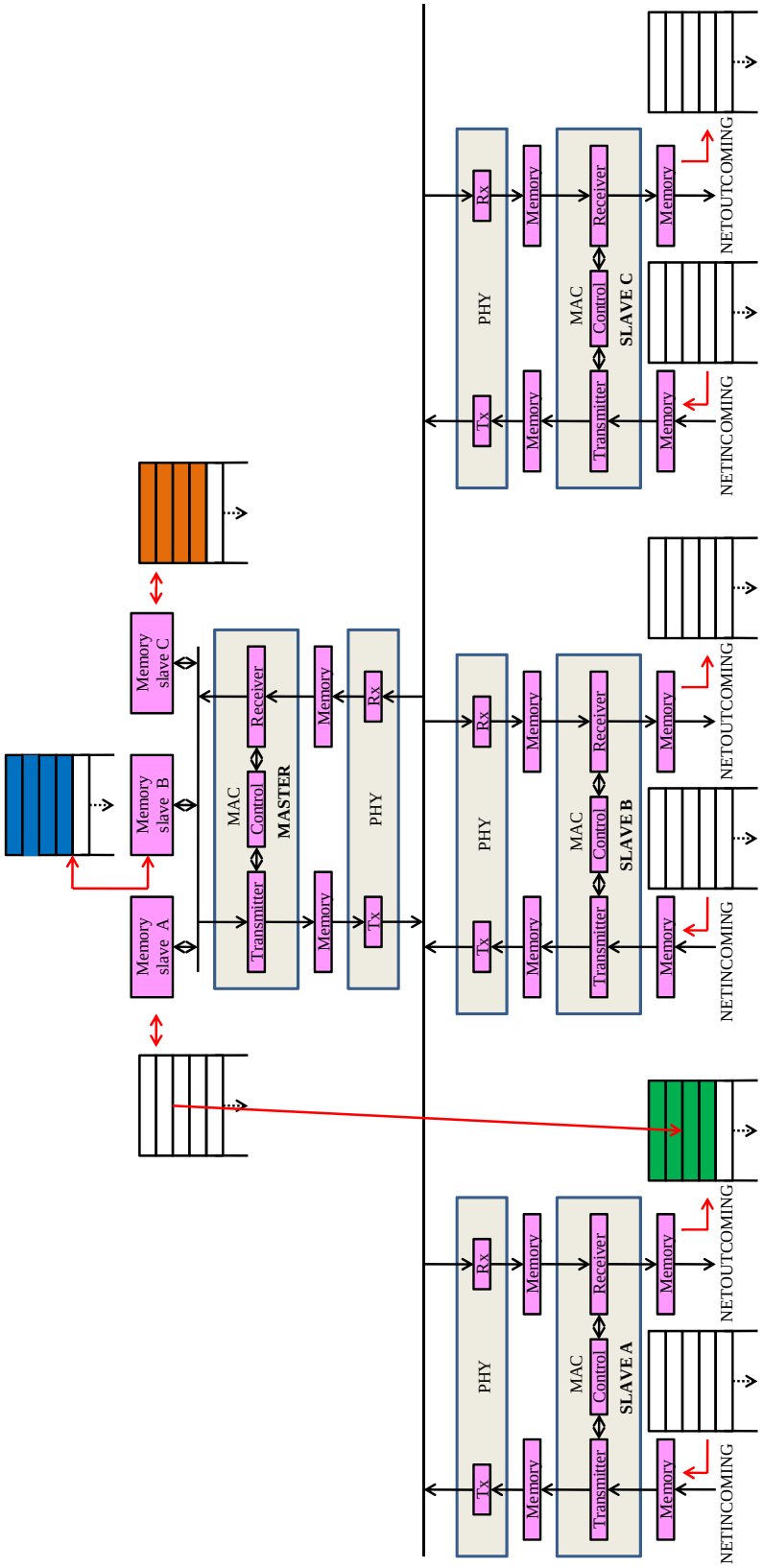

Fig 10.6 Second verification: Master polls Slave A with downlink connection

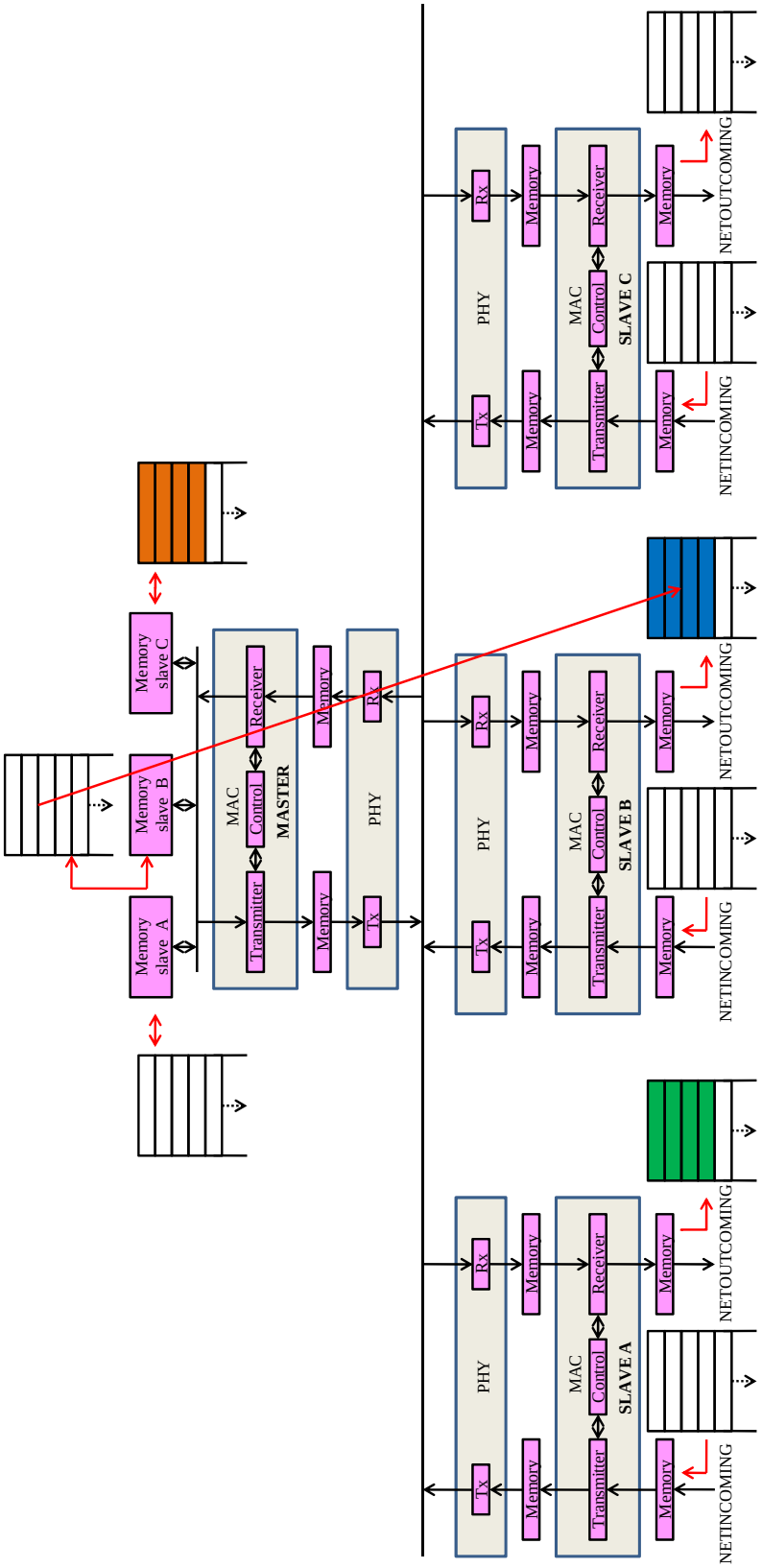

Fig 10.7 Second verification: Master polls Slave B with downlink connection

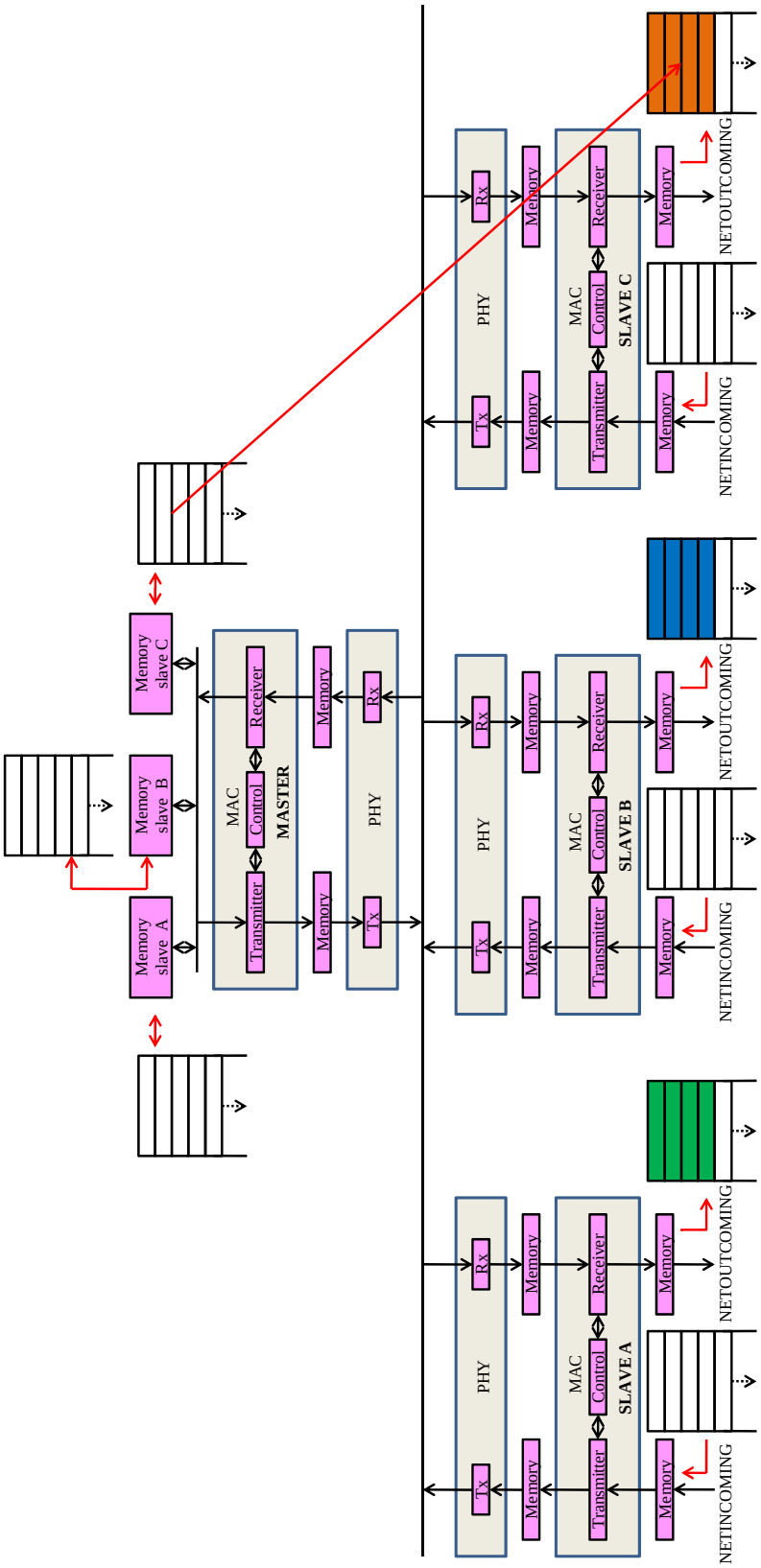

Fig 10.8 Second verification: Master polls Slave C with downlink connection

10.3 MASTER CONTROL BLOCK STATE MACHINE

The complete state machine of the master control block is shown in the fig 10.9. It contains the uplink and downlink mode.

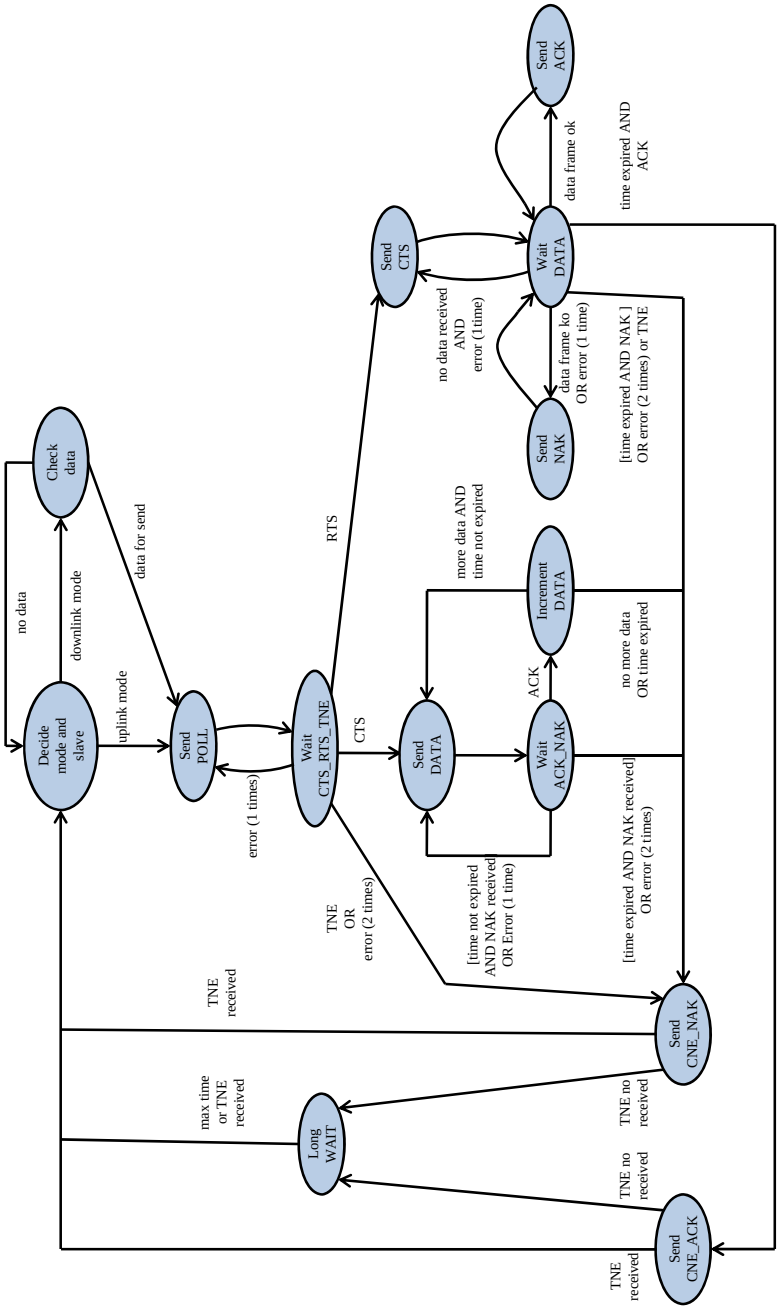

Fig 10.9 Complete state machine of the Master Control Block

10.4 SLAVE CONTROL BLOCK STATE MACHINE

The complete state machine of the slave control block is shown in the fig 10.10. It contains the uplink and downlink mode.

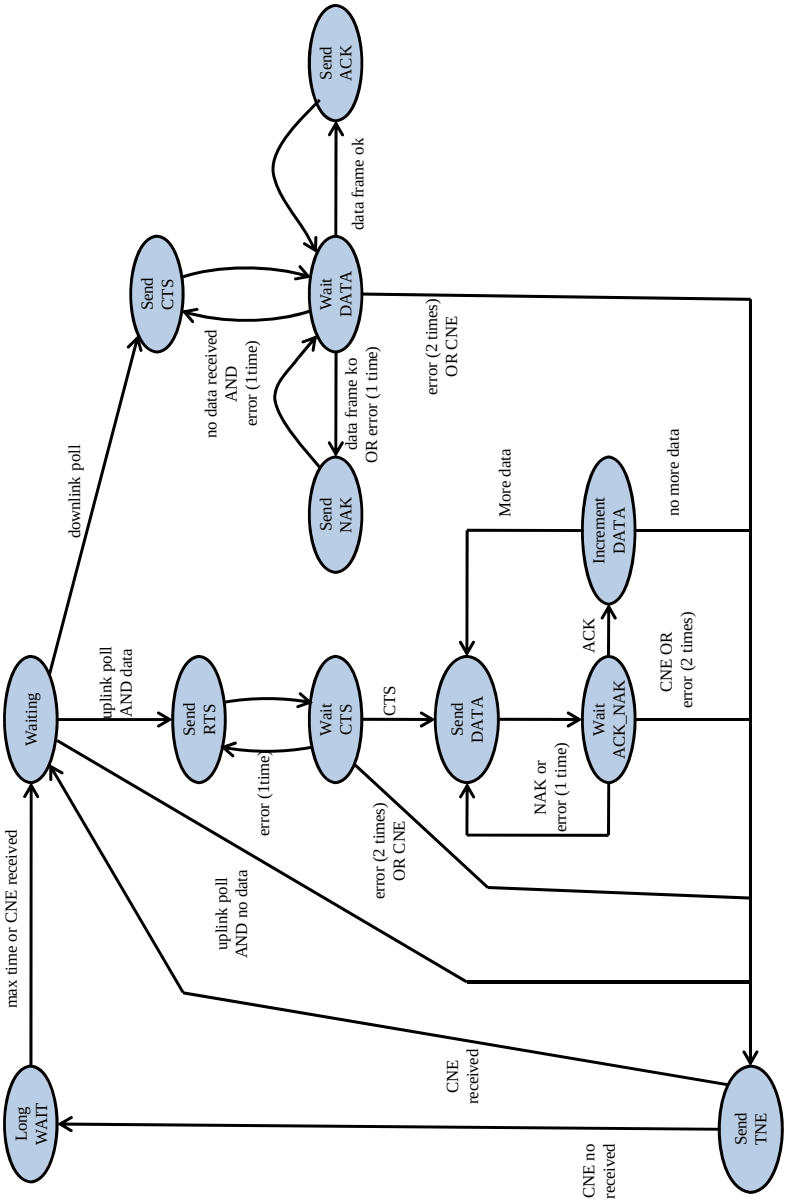

Fig 10.10 Complete state machine of the Slave Control Block

References

Theoretical part

Kurose, James F. “Computer networking : a top-down approach”, 3th Ed, Boston, Mass. ; Munich [u.a.] : Pearson Addison-Wesley, 2005

Tanenbaum, Andrew S. “Computer networks” 4th Ed, Upper Saddle River, NJ : Pearson Education International, 2003

W. Stallings “Comunicaciones y Redes de Computadores” 6th Ed, Prentice Hall, 2001

Leon- Garcia, Alberto “Communication networks : fundamental concepts and key architectures” 2nd Ed, Boston [u.a.] : McGraw-Hill

http://www.altera.com/literature/an/an049_01.pdf

VHDL

Peter J. Ashenden “The system designer's guide to VHDL-AMS” Amsterdam [u.a.] : Kaufmann, 2007.

Yu-Chin Hsu “VHDL modeling for digital design synthesis” Boston [u.a.] : Kluwer, 1995

http://diec.unizar.es/~tpollan/libro/librodigital.htm

Cyclone III fpga starter board

http://www.altera.com/products/devkits/altera/kit-cyc3-starter.html#ordering

ftp://ftp.altera.com/outgoing/devkit/91/cycloneIII_3c25_start_v9.1.0.exe

Modelsim

http://portal.model.com/modelsim/resources/references/m_qk_guide.pdf

http://portal.model.com/modelsim/resources/references/modelsim_se_user.pdf

http://portal.model.com/modelsim/resources/references/modelsim_se_ref.pdf